
MAGAZINE
MSX
AÑO I
Núm. 2
250 Ptas.
MSX-DOS, un sistema operativo de verdad
Sinfonía de sonidos
Desnudamos el hardware MSX
Bases numéricas para todos
ANDRES SANCHEZ

¡EL IMPERIO CONTRAATACA!

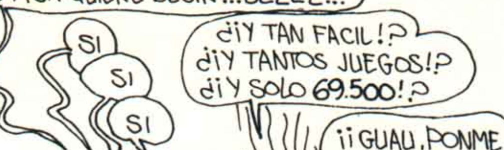












Ordenador Personal
TOSHIBA HX-10
Su Ordenado Servidor

69.500 Ptas.

MSX SYSTEM



Características principales:
Sistema standard MSX. Memoria de 64 K RAM, 32 K ROM y 16 K de pantalla. 16 colores. 73 teclas. 32 sprites. Sistema multicolor: 64 x 48 bloques. Sonido: 8 octavas tres acordes. Conexiones para: casette, impresora, 2 mandos y futuras expansiones.



TOSHIBA

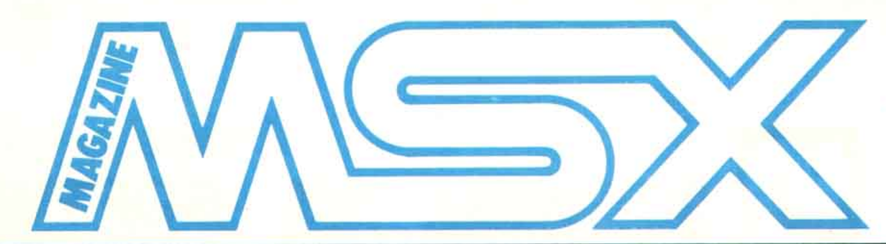
española de microordenadores s.a.

Caballero, 79 - Tel. 321 02 12 - Telex 97087 EMOS - 08014 BARCELONA

MSX SYSTEM

El sistema MSX es un standard utilizado universalmente que permite disponer de una gran variedad de programas y accesorios compatibles entre sí.

PUZZLE



AÑO I - Núm. 2
Junio 1985

La acogida del primer número de MSX Magazine ha sido ejemplar. Ello prueba la necesidad de cubrir un hueco dejado en el mercado español dedicado a este singular estándar. No cabe la menor duda de que ha nacido una nueva época para el ordenador personal y nosotros con ellos.

En este número que tiene en sus manos hablaremos de temas interesantes como, interioridades del ordenador, generación de sonidos, etcétera. Además, en las páginas interiores invitamos a los lectores a que envíen sus programas, con posibilidad de publicarlos en la revista, manteniéndonos firmes en el propósito de hacer que MSX Magazine sea el fruto directo de la estrecha colaboración entre lector y redacción.





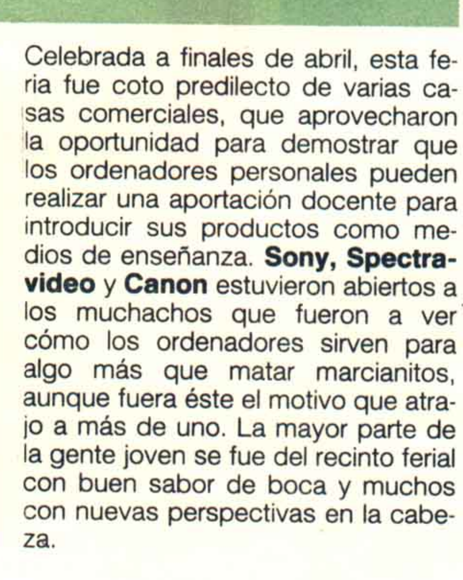


NOTICIAS



SOFTWARE EUROPEO

IBERDIDAC '85



Celebrada a finales de abril, esta feria fue coto predilecto de varias casas comerciales, que aprovecharon la oportunidad para demostrar que los ordenadores personales pueden realizar una aportación docente para introducir sus productos como medios de enseñanza. **Sony, Spectravideo** y **Canon** estuvieron abiertos a los muchachos que fueron a ver cómo los ordenadores sirven para algo más que matar marcianitos, aunque fuera éste el motivo que atrajo a más de uno. La mayor parte de la gente joven se fue del recinto ferial con buen sabor de boca y muchos con nuevas perspectivas en la cabeza.

TOSHIBA: GUERRA DE PRECIOS

El mes pasado, **Toshiba** lanzaba al mercado el ordenador MSX HX-10 al precio de 69.500 pesetas. Este mes, sin embargo, las noticias son inmejorables, sobre todo para el que deseaba poseer un MSX y argüía que los precios eran demasiado altos.

La constante evolución del mercado de los ordenadores personales está haciendo su efecto. Ya se han iniciado, por parte de varias casas comerciales, campañas de publicidad en televisión y en diversos medios de comunicación, pero ninguno a jugado la baza de bajar los precios. Sin lugar a dudas, este será el mejor argumento de ventas de estos ordenadores, que han visto reducido su precio en casi 10.000 pesetas.

Sí, efectivamente, el nuevo precio que entrará en vigor este mes es de 59.900 pesetas y, además, con la adquisición de estos ordenadores se incluye un cassette para aprender el BASIC MSX. Por si esto fuera poco, para este mes también se anuncia la llegada de los ansiados discos de esta marca. Toda una serie de innovaciones que calentará, más aún, la llegada del verano 85.

Una firma holandesa **Aackosoft,** están desarrollando *software* especialmente dedicado para aplicaciones y enseñanza, comercializándolos en países como el Reino Unido, Francia y Alemania. Pionera en su país de los "campamentos informáticos", en labor conjunta con **Sony** está obteniendo un auge importante de la industria del ordenador personal en aquel país, gracias a enseñar a la juventud las posibilidades de estos aparatos. Poco falta, pues, para que esta importante firma inicie la comercialización de sus programas, que incluye desde un tratamiento de textos hasta un generados de *sprites,* en nuestro país.

SPECTRAVIDEO: RED LOCAL PARA USOS DOCENTES

Sin lugar a dudas, la más importante innovación en MSX la constituye la presentación, por parte de **Indescomp,** de la red local de Spectravideo.

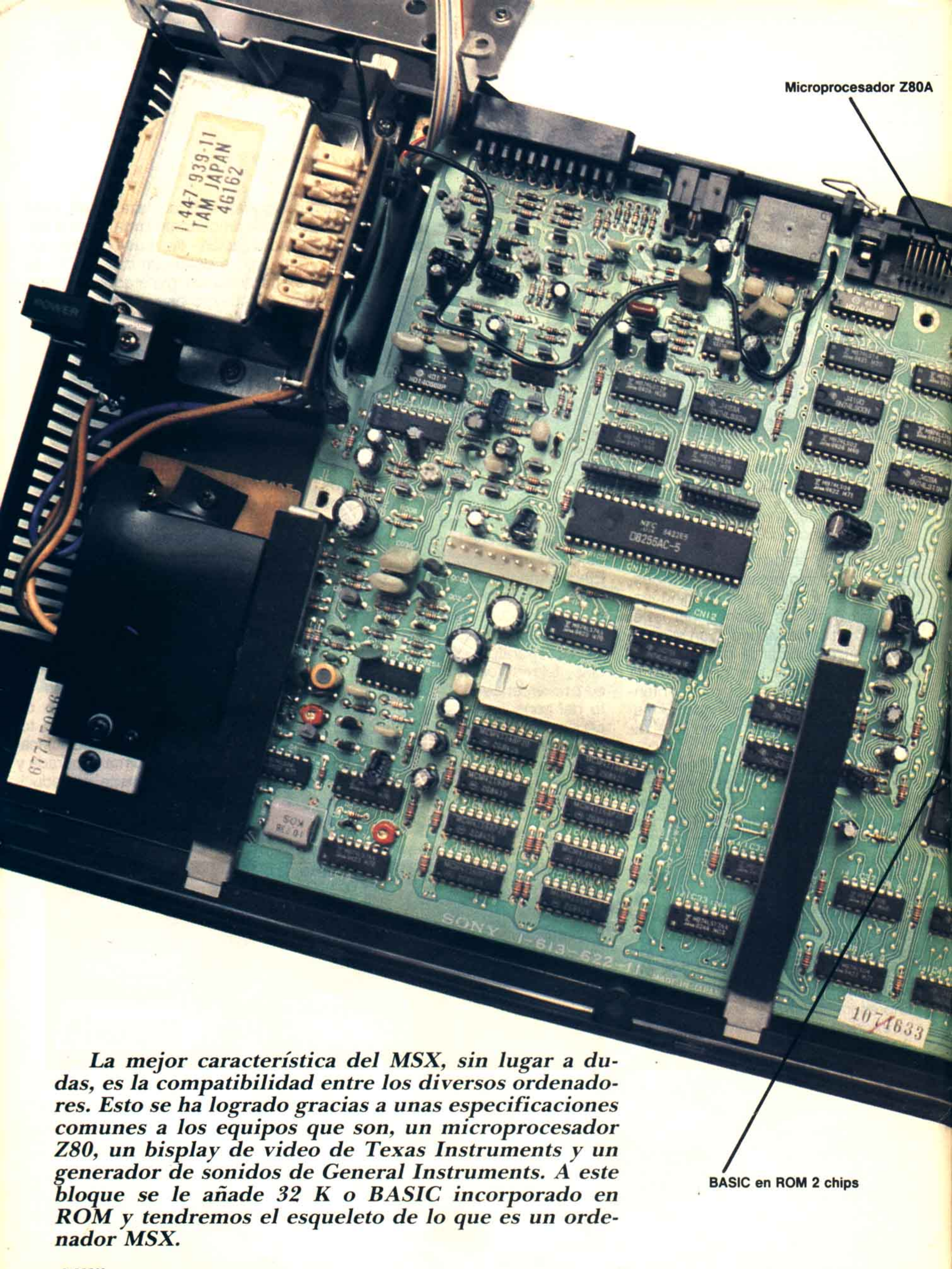
La rueda de prensa que fue presidida por el propio presidente de Indescomp, señor José Luis Domínguez, se realizó en los salones del hotel Meliá Castilla de Madrid. Con sus palabras hizo constar la preocupación de la entidad por sus grandes vías comerciales de actuación, la potenciación del nuevo ordenador Amstrad y el desarrollo de su red local de comunicaciones MSX.

Con la posibilidad de poder conectar un total de 32 MSX, el único requisito indispensable es que éstos posean al menos 64K de memoria RAM. Las posibilidades de esta red aumentan considerablemente al ir todos los ordenadores conectados a una estación maestra con las siguientes características: partiendo del SV-328 y un expansor de red local, con un disco Winchester de 10 Megabytes formateados, disco *floppy* de 320K formateados, *interface* Centronics para impresora, *interface* serie RS-232C, ampliación de 64K RAM y tarjeta de 80 columna.

Nuestro Expreso de Oriente no se ha dormido en los laureles, y nos ha traído unas pequeñas noticias curiosas del otro extremo del planeta. Dos son las curiosidades que vienen a nuestras páginas, la primera de ellas es la próxima aparición de robot MSX. Efectivamente, una importante empresa japonesa ha desarrollado un robot que, conectado a un ordenador MSX, permitirá controlar una serie de aparatos como pletinas dobles, televisores, videos, etc. Estos se podrán encender y apagar a gusto del consumidor, puesto que será él quien programe su ordenador para que éste realice todas estas tareas. Este es el robot doméstico, mientras que en el otro extremo, están pensando en una versión para usos industriales que la puede revolucionar.

La otra noticia hace referencia al complicado entramado interior de los MSX. Parece ser que en un futuro no muy lejano y en vistas de la pronta aparición de los MSX de procesadores de 8/16 *bits,* la presentación interna de los aparatos va a cambiar. La mayoría de estos ordenadores vienen con gran cantidad de circuitos integrados y 8 *chips* separados para la memoria de 64K. Esto está en vías de desaparición, ya que se piensa reducir y agruparlos en un *chip* estandard. El nuevo *chip* de memoria, MT8064 almacenará la totalidad de los 64K. Este revolucionario *chip* tendrá una forma cuadrada y sólo 36 *pins.* Obviamente, se reducirá sensiblemente el consumo, aumentando las prestaciones del ordenador.





La mejor característica del MSX, sin lugar a dudas, es la compatibilidad entre los diversos ordenadores. Esto se ha logrado gracias a unas especificaciones comunes a los equipos que son, un microprocesador Z80, un bisplay de video de Texas Instruments y un generador de sonidos de General Instruments. A este bloque se le añade 32 K o BASIC incorporado en ROM y tendremos el esqueleto de lo que es un ordenador MSX.

MSX: Interioridades

Generador de Sonido PSG

El conocer las interioridades de un ordenador nos permitirá un mejor entendimiento de las posibilidades del aparato a todos los niveles, no sólo, en cuanto a su capacidad de tratar la información, sino también, a la posibilidad de interconectarlo con periféricos y aparatos que estaban reservados para otro tipo de ordenador más específico. Este conocimiento nos permitirá aprovechar nuestra máquina y "exprimirla" hasta el último *bit* de memoria. El **MSX** tiene la particularidad de poseer chips con funciones totalmente independientes, no como otros ordenadores, donde la CPU realiza las funciones de microprocesador, generador de sonido y display de video.

En esta introducción, mostraremos paso a paso las características de las partes principales del ordenador y de otras partes que aunque no sean de vital importancia, el papel que realizan sí es crucial, como, por ejemplo, el registro que selecciona las participaciones de memoria. Esta cuestión viene descrita en los manuales que nor-



malmente acompañan al ordenador, pero en el mejor de los casos, sólo lo pueden entender los que poseen conocimientos de electrónica. Nosotros procuraremos ponernos a buena altura para perder ese miedo que da el desconocimiento de semejantes máquinas y hacerle comprender las características esenciales de su aparato. Antes de entrar en materia, conviene aclarar un símbolo que se repetirá hasta la saciedad en este y otros artículos, se trata del *"&"* que junto con la letra *"O"* o *"H"* indican la base numérica en la que se está trabajando. El primero se utiliza para operar con números en base octal y el segundo para operar en base hexadecimal.

La memoria RAM es la herramienta adicional de la ROM, haciendo aquélla las funciones de calculadora y bloque de anotaciones.

Iniciaremos este estudio con el mapa de memoria de la figura 1, en el que podemos observar la aparición de un concepto totalmente nuevo para algunos y no tanto para otros, se trata de la partición, que más adelante veremos con detalle, por el momento basta saber que la partición **0** es la del sistema. El mapa de memoria de un ordenador es una descripción de como está repartida y aprovechada ese espacio que es la memoria. A su vez también podremos comprobar el estado de la zona dedicada a la Entrada/Salida de información. La *ROM* del **MSX** siempre comienza en la dirección **0** de la memoria y ocupa hasta la dirección *&H7FFF*, mientras que la *RAM* ocupa desde la dirección *&H8000* hasta la *&HFFFF*. Un equipo **MSX** necesita una configuración mínima de 8K, pero esto limita enormemente las posibilidades del sistema. Estos 8K ocupan la parte inferior de la memoria, es decir, desde la dirección *&HFFFF* hacia abajo. Aunque esta sea la configuración mínima, se le pueden añadir bloques de *RAM* de 16K, que ocuparían desde la dirección &HFFFF hasta la *&HC000*. Haciendo una resta en hexadecimal (en otro momento hablaremos de las operaciones con los distintos sistemas numéricos, principalmente los sistemas binarios, octal y hexadecimal) obtendremos como resultado 16384 posiciones de memoria. Esto sorprenderá a más de uno, ya que si añadimos 16K a los 8K existentes, deberíamos obtener un total de 24 K, y esto no es cierto, ya que los 8K son absorbidos por los 16K que añadimos. La razón es que el MSX trabaja con bloques de información de 16K denominado página. Ahora estamos en condiciones de hacer una afirmación a partir de la cual continuaremos explicando el funcionamiento de la memoria en este sistema. Un sistema MSX con una memoria mínima de 16K, posee una página de *RAM* que ocupa las direcciones *&HCFFF* hasta la *&HFFFF* y dos páginas de *ROM*, que ocupan están integrados el BASIC junto con el OS (Operating System-Sistema operativo). Hasta el momento no hemos visto nada espectacular en el interior del aparato, salvo el concepto de la partición. Este marca la pauta diferenciadora entre ordenadores personales.

Memoria ROM

El estándar **MSX** la trata por particiones, que es una de las características que permite añadir memoria a la ya existente, aunque esta posea *32K* de *RAM* y *32K* de *ROM*. Con una memoria de este tipo, tendremos un completo mapa de memoria, ya que el microprocesador Z80 sólo puede manejar 64K a la vez. En resumen, el ordenador puede tomar particiones indepen-

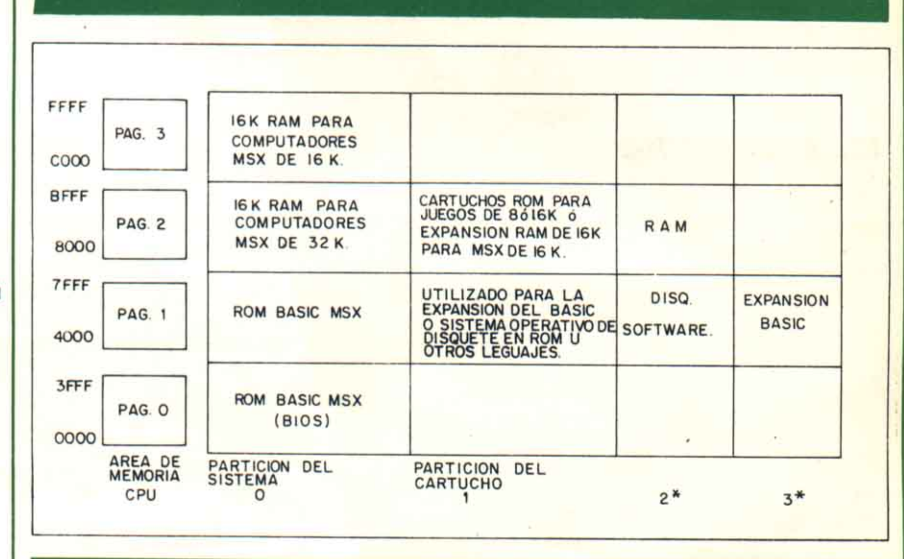
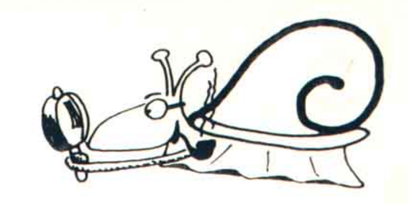


Figura: 1
Mapa de memoria de las características del MSX.



dientes para formar el mapa de memoria. Por lo mencionado podemos deducir que una partición tiene un total de 64K (4 páginas de 16K = 65535 bytes) que se puede rellenar de *ROM* o *RAM*. Absolutamente todos los ordenadores **MSX** han de poseer por lo menos dos particiones de memoria, aunque se puedan tener hasta 4. Cada una de estas particiones se pueden expandir en cuatro más, por lo que el total de particiones que podemos tener es de 16. Es más, si cada una de éstas tuviera 64K, algo que no se puede conseguir con otros ordenadores personales; la posibilidad de direccionar hasta 1 Megabyte de memoria. Esta es la cantidad máxima tolerada por el **MSX**,

El interface de comunicaciones 8251 y el programador de intervalos 8253 son los encargados de las conexiones de E/S.

aunque sólo se puedan manejar 64K a la vez. Las particiones suelen tener una función determinada. La primera partición es la *0* y es la del sistema, donde tendremos *32K* de *ROM* y *16K* de *RAM*. La siguiente partición es la 1 o la del cartucho *ROM*, donde podremos conectar ampliaciones de memoria de 16K o cartuchos de juegos en ROM.

Las particiones son controladas por el registro seleccionador que es el *port A* del *8255 PIA*. Este registro informa al ordenador sobre las particiones a tener en cuenta a la hora de realizar el mapa de memoria. La figura 2 muestra la forma en que se acceden a las particiones de la memoria. Los bits 0 y 1 acceden a páginas situadas en la partición 0, los bits 2 y 3 acceden a las páginas de la partición 1 y así hasta la partición 4, a la cual se accede con los bits 6 y 7. Hay que hacer hincapié sobre la necesidad de tener nociones de números binarios, ya que, la memoria de cualquier ordenador trabaja en este sistema de numeración. Aquí es binarios del 0 al 3). Hay que destacar que la memoria física está de alguna forma absorbida por la página específica y de esa forma, mientras que es posible acceder a la página 2 desde la partición 3 como parte del mapa de memoria, no se puede asignar la página 3 de la partición 3 a la página 0 del espacio de memoria de la *CPU*.

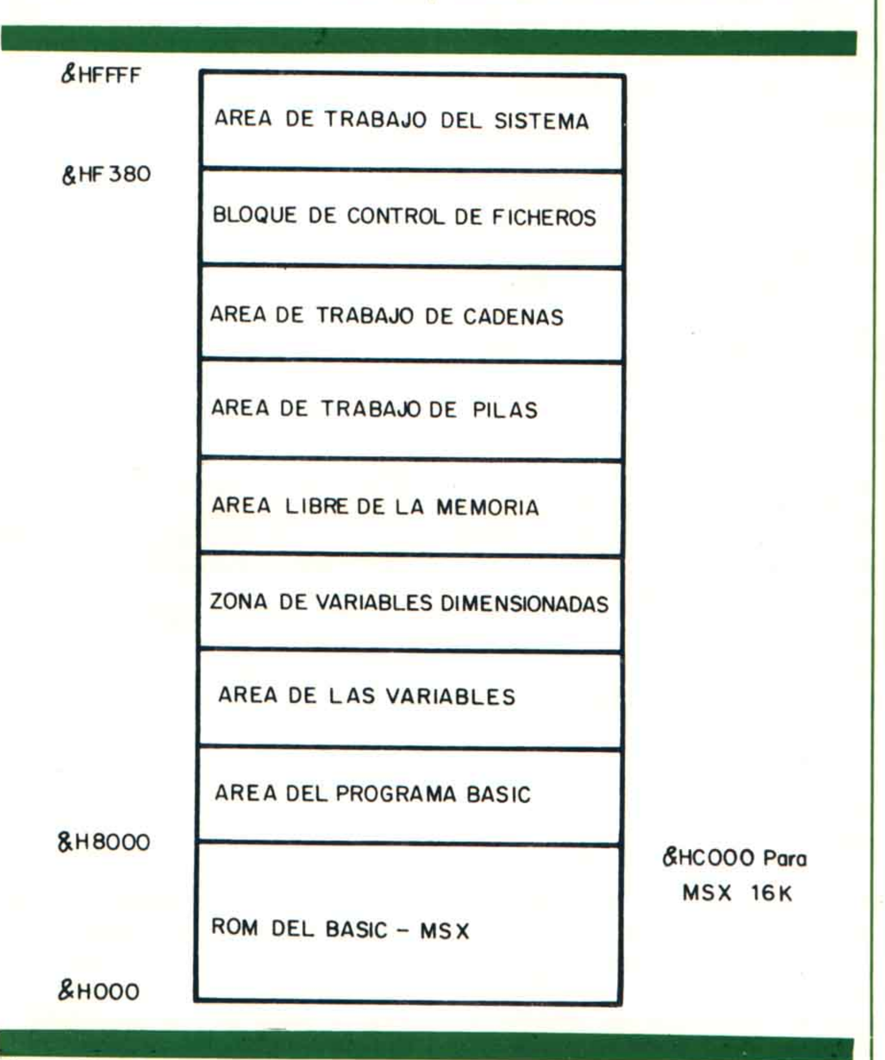


Figura: 1 **bis;**
Mapa de memoria de la partición del sistema.

útil saberlo, puesto que el contenido de este registro son 0 y 1 que según su combinación, va a dar el valor de una de las cuatro páginas de la partición. En este registro los números se representan de la forma 00, 01, 10 y 11 (son los números

Una mínima configuración de un sistema ha de llevar dos particiones, una para el ordenador y otra para el cartucho. Normalmente, las instrucciones que acompañan a cualquier **MSX** confunde un poco al usuario con el término de

partición, que es la forma en que está dividida la memoria del ordenador. De esta forma, mientras que una partición controla un cartucho, es a su vez una partición conceptual y físicamente, la partición del ordenador no tiene ningún sentido físico.

La partición 0, que es la del sistema, está dividida en una serie de áreas de trabajo que resultan conveniente conocer, estas son las siguientes. Una área de trabajo, la cual está situada en las siguientes zonas de memoria, desde la *&HFFFF* hasta la *&HF380* y es utilizada por la memoria *ROM* del ordenador para realizar sus operaciones y sus cálculos internos. La siguiente zona a saber es el Bloque de Control de Ficheros, que está reservada para las operaciones de Entrada/salida cuando utilicemos ficheros. Las instrucciones como *PRINT#* e *INPUT#* utilizan este bloque. El tamaño del bloque se puede fijar con la instrucción *MAXFILES*. El límite máximo está situado en la dirección *&HF380*, pero se puede controlar con la instrucción *CLEAR*. El área de las cadenas, la cual almacena los contenidos de las variables de cadena, si no se especifica este valor será de 200 caracteres. Luego tenemos la zona de las Pilas. Quizás la más interesante, puesto que conociéndola, tendremos un control sobre los bucles *FOR-NEXT*. Aquí se almacena la dirección de la instrucción BASIC a la que hay que volver una vez ejecutado el bucle. La siguiente área de la memoria es una que no se utiliza y cuyo tamaño lo podemos comprobar con la instrucción *FRE* desde el BASIC. Esta instrucción nos dirá la capacidad de nuestro ordenador. Por ejemplo, introduciendo el siguiente comando directo, sabremos la memoria de nuestro ordenador. *PRINT FRE (0)*, dará un valor que nos indicará si nuestro ordenador es de 32 o 64K o si es de 16K, en el primer caso obtendremos *28815* y en el segundo caso obtendremos 12431. El tamaño disminuye en función de la longitud de nuestro programa BASIC. Acabando con esta primera partición tenemos tres áreas que son las que controlan las variables dimensionadas, las variables propiamente dichas y la zona del programa BASIC. La primera almacena las variables de una instrucción DIM y se incrementará cada vez que ejecutemos esa instrucción. La segunda almacena las variables numéricas y la cadena de punteros de esta zona. Por último, encontraremos la zona dedicada a almacenar un programa en BASIC.

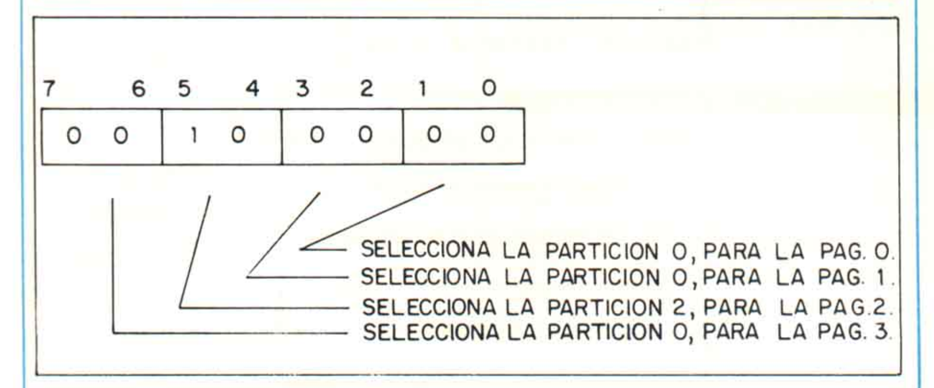


Figura: 2
Registro de selección de las particiones del mapa de memoria.

Conectores Entrada/Salida

Pasemos a ver un poco los conectores de Entrada/Salida de información. En la figura 3 podemos ver como las 256 direcciones de E/S del procesador Z80 están distribuidas dentro de un ordenador **MSX.** Veremos detenidamente lo que indica este mapa de memoria, comenzando por el interface RS-232 en las direcciones *&H80* y *&H88*.

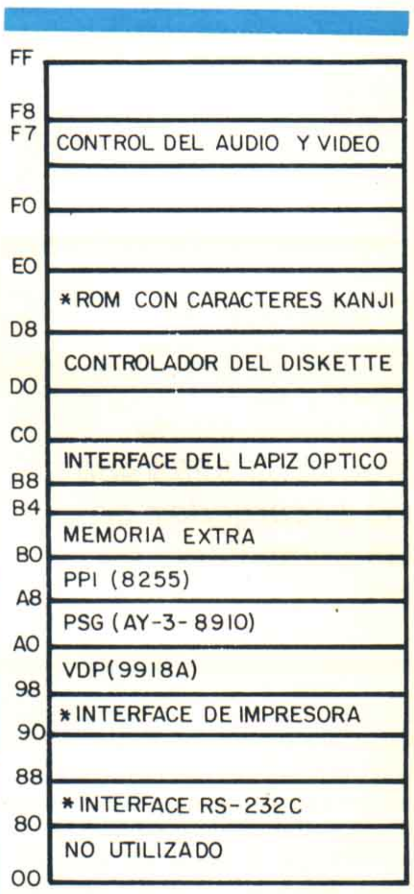
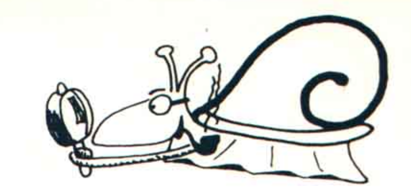


Figura: 3
Mapa de memoria de E/S.

Está basado en un chip interface de comunicaciones tipo 8251 y un programador de intervalos IC 8253. La asignación de ports del RS 232 dentro del área de memoria del mapa de E/S son las siguientes:

** &H80-L/E 8251 port de datos,



** &H81-8251 port controlador del estatus de los comandos,

** &H82-L interruptores que controlan la velocidad en baudios,

** &H83-L interruptores que controlan la configuración,

** &H84-E máscara del registro de interrupción,

** &H83- L/E 8253 contador 0,

** &H83- L/E 8253 contador 1,

** &H86- L/E 8253 contador 2,

** &H87- E 8253 registro de modos.

Los interruptores de las direcciones *&H82* y *&H83* son las responsables de la configuración del canal RS-232. &H82 prepara la velocidad en baudios, donde los bits 0-3 controlan la entrada de datos y los bits 4-7 controlan la

La configuración mínima de un ordenador MSX es de 8K, pero esta posibilidad delimita enormemente las posibilidades del aparato.

velocidad de transmisión. Fijando el valor adecuado con número hexadecimal entre *&H0* y *&HA*, podremos tener acceso a las siguientes velocidades: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600 o 19200, todos estos valores están expresados en baudios. Los valores entre *&HB* y *&HA*, podremos tener ac- *&HB* y *&HE*, están sin asignar, mientras que *&HF* es un canal que no se utiliza. La direccíçn *&H83* se ocupa del resto de la configuración de los canales. El bit 0 es la línea CD conectada directamente con el RS-232 mientras que los restantes 7 bits realizan la siguiente función. El bit 1 selecciona la alimentación automática de línea, el bit 2 controla si la línea es *full* o semi duplex (término en la transmisión de datos que veremos más adelante). El bit 3 determina si el control XON/OFF está conectado y el bit 4 controla la longitud de la palabra y el bit 5, fija la paridad, que puede ser par o impar (este concepto también lo veremos detalladamente, ya que es un punto fundamental en la transmisión de datos). Bit 6 pone la paridad si el valor es alto y el bit 7 fija los bits de stop, 2 si se pone y 1 si no.

Otras áreas del mapa de E/S no preocuparán a la mayoría de los usuarios, a excepción del *port PPI*. Este está colocado de la siguiente forma;

** &HA8- Port A de E/S,

** &HA9- Port B de E/S,

** &HAA- Port C de E/S,

** &HAB- Modo del registro.

Aunque el manual comenta las direcciones de E/S, advierte que algunos fabricantes a lo mejor no siguen las especificaciones con toda exactitud. De manera, que para evitar problemas ue esto pudiera causar, la mejor forma de controlar todas las llamadas a los *ports* de E/S es mediante el uso de las llamadas al sistema operativo.

Modo		Resolución	Tamaño	Núm.	Colores	Sprites	Núm. de caract.
Gráfico I	LSI esperado	256 × 192	8 × 8	256	16 colores	SI	32 × 24
	Valor sugerido	240 × 192					29 × 25
Gráfico II	LSI esperado	256 × 192	8 × 8	768	16 colores	SI	32 × 24
	Valor sugerido	240 × 192					29 × 34
Multicolor	LSI esperado	64 × 48 Bl.	Bloq.		16 colores		32 × 24
	Valor sugerido	64 × 40 Bl.	4 × 4	—	—	No	29 × 24
Texto	LSI esperado	256 × 192	8 × 6	256	2 colores de los 16 posibles	SI	40 × 24
	Valor sugerido	240 × 192					39 × 24

Figura: 4
Configuración de la partición de cartuchos.

Siguiendo con la explicación de las diversas zonas de memoria llegamos a las siguientes;

** &H90 a &H91- Esta controla el interface de la impresora, por lo que es utilizado por los comandos del BASIC, *LPRINT* y *LLIST*. El bit 1 de la dirección *&H90* indica si la impresora está o no ocupada. Todo lo que escribamos en la dirección &H91 saldrá por la impresora.

** *&HA0* a *&HA2*- Son direcciones muy útiles pues permiten controlar el chip programable de generación de sonido (PSG, AY-3-8910) a través de las instrucciones *INP* y *OUT*. Se puede controlar

directamente al programar en código máquina, pero es más sencillo en BASIC ya que se puede controlar mediante las citadas instrucciones.

** &H99- Estas dos direcciones permiten un acceso directo a la memoria del video display (*VDP, 9918A*) sin tener que utilizar las instrucciones de este chip. También permite acceder directamente a la *VRAM* sin tener que utilizar las instrucciones *VPEEK* ni *VPOKE*

** *&HA8* a *&HAB*- Llegamos a una zona de memoria que hará las delicias de unos, pero que sumirán en un mar de dudas a otros, se trata del interface programable de periféricos. Este hay que manejarlo con sumo cuidado ya que podemos causar serios problemas y más si tenemos algún programa en la memoria del ordenador.

Nuestra sugerencia es que no se utilice a no ser que se domine el tema. Este interface es el chip *PPI 8255* y está compuesto de cuatro registros, cuya disposición es la siguiente;

— &HAB, registro A de salida
— &HA9, registro B de entrada
— &HAA, registro C de salida
— &HAB, registro para seleccionar el modo.

Registros

Este interface permitirá controlar cassettes, teclados, interfaces electrónicos, etc. Para un mejor entendimiento, veamos los registros individualmente.

El registro A es de salida y se utiliza para controlar las direcciones de memoria del sistema **MSX**. Su contenido no se debe alterar a no ser que sepamos lo que hacemos.

El registro B es de entrada y detecta si se ha pulsado una tecla. El registro C es de salida y su función primodial es la de ayudar en la lectura del teclado, función que por el momento no nos interesa y que en futuros números no solo comentaremos sino que dedicaremos especial atención a este interface programable de periféricos.

** *&HBO* a *&HB3* - Se utiliza en algunos ordenadores para controlar memoria adicional.

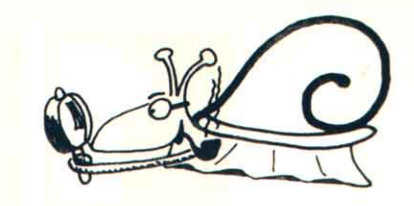
N.° PIN	NOMBRE	E/S	N.° PIN	NOMBRE	E/S
1	CS1	S	2	CS2	S
3	CS12	S	4	SLTSL	S
5	RESERVADA	—	6	RFSH	S
7	ESPERA	E	8	INT	E
9	M1	S	10	BUSDIR	E
11	IORQ	S	12	MERQ	S
13	WR	S	14	RD	S
15	RESET	S	16	RESERVADA	—
17	A 9	S	18	A15	S
19	A11	S	20	A10	S
21	A7	S	22	A6	S
23	A12	S	24	A8	S
25	A14	S	26	A13	S
27	A1	S	28	A0	S
29	A3	S	30	A2	S
31	A5	S	32	A4	S
33	D1	E/S	34	D0	E/S
35	D3	E/S	36	D2	E/S
37	D5	E/S	38	D4	E/S
39	D7	E/S	40	D6	E/S
41	GND	—	42	RELOJ	S
43	GND	—	44	SW1	—
45	+5V	—	46	SW2	—
47	+5V	—	48	+12V	—
49	E/SONIDO	E	50	—12V	—

Figura: 5
Diagrama de conexiones en los cartuchos ROM.

N.° PIN	NOMBRE SEÑAL
1	PSTB
2	PDB0
3	PDB1
4	PDB2
5	PDB3
6	PDB4
7	PDB5
8	PDB6
9	PDB7
10	NC
11	OCUPADO
12	NC
13	NC
14	GND

Figura: 6
Diagrama de las conexiones que conforman el port de la impresora.

La partición 0, que es la del sistema, está dividida en áreas de trabajo con tareas específicas.



** &HBB a &HBB - Esta dirección es caso aparte puesto que controla el lápiz óptico que algunos ordenadores tienen como opción. Con esta dirección finaliza un estudio previo de las direcciones de memoria de un **MSX**, de cualquier manera, el ordenador tiene bastantes elementos a los que dedicaremos ríos de tinta hasta que los lectores hayan comprendido su completo funcionamiento.

N.° PIN	NOMBRE SEÑAL	DIRECCION
1	AVANCE	ENTRADA
2	RETROCESO	»
3	IZQ.	»
4	DER.	»
5	+5V	—
6	DISP. 1	ENTRADA
7	DISP. 2	SALIDA
8	SALIDA	SALIDA
9	GND	—

Figura: 7
Asignación del port de entrada.

Memoria RAM

Continuando con las interioridades del aparato, entremos en la disposición de la memoria RAM.

Como ya sabréis, la memoria RAM es el lugar donde se almacenan los programas en BASIC y las variables de ese programa. Empezaremos por ver la dirección inicial de un programa. Para hallar dicha dirección tenemos que introducir el siguiente comando directo:

PRINT 65536 - (n.° 1024)
donde n es el número de K de tu ordenador.

Claro que el ordenador necesita de espacio para trabajar y de alguna parte lo ha de obtener. En el **MSX**, ordenador complicado de por sí, se necesita utilizar áreas de la memoria *RAM* denominada **Area de Trabajo del Sistema** y por esa razón al conectar el aparato no tienes toda la memoria que uno se cree, ya que antes de poder teclear tus programas, el ordenador se asegura una cierta zona que le permita realizar sus operaciones. La zona que reserva está ubicada a partir de la dirección 62336, por lo que hay que poner especial atención a la hora de realizar algún *POKE* en esta zona de la memoria. La dirección del comienzo del área de trabajo del sistema la podemos hallar en las direcciones 64586 y 64587, el inicio de esta zona la podemos calcular de la misma manera que cuando calculamos la dirección inicial de *RAM*. Otra zona dentro del área de trabajo del sistema es la que se dedica al reloj. Esta variable, llamada TIME, se encuentra en las direcciones 64670 y 64671 y se incrementa con regularidad. Estos son algunos de los elementos dentro de esta zona especial para el área del sistema. Hay más, pero su complicada estructura hacen innecesario su descripción por el momento.

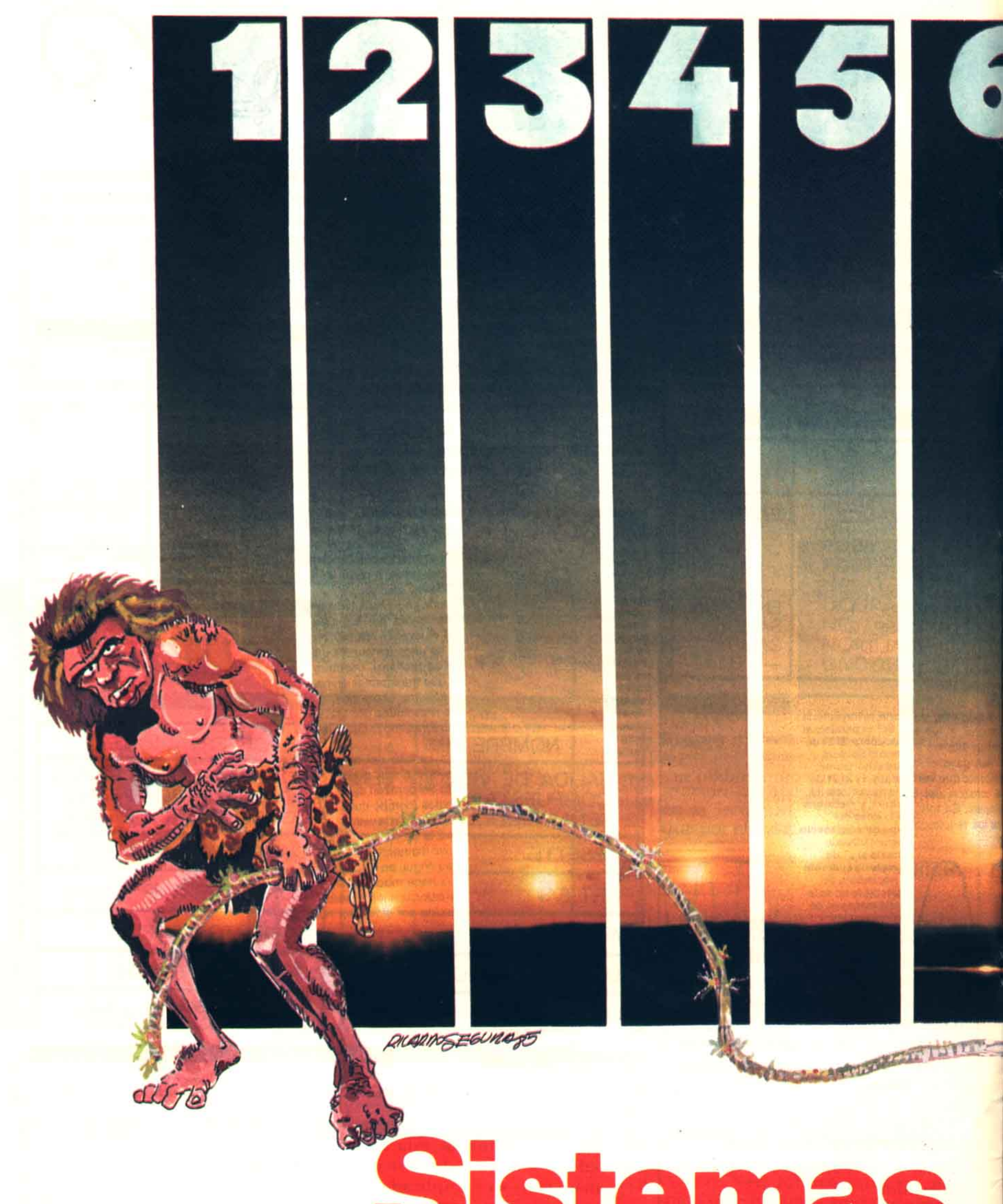
Una vez vistas las interioridades más relevantes del **MSX**, sería útil que los lectores observaran las figuras de la 5 a la 9, donde mostramos la disposición de los pines de los conectores como la de los cartuchos (figura 5), el *port* de la impresora (figura 6), *port* analógico de entrada de datos (figura 7), el port de Entrada/Salida del cassette y por último una especificación de los conectores e interfaces utilizados en los ordenadores **MSX** (figura 9). Aquí ponemos punto final a esta pequeña introducción sobre las interioridades del sistema. Es conveniente que se prueben todos los conceptos vistos hasta el momento y repasarlo, para que de esa forma os resulte fácil y divertido dominar vuestro ordenador y no dejaros dominar por él. □

N.° PIN	NOMBRE SEÑAL	DIRECCION
1	GND	—
2	GND	—
3	GND	—
4	CMT SALIDA	SALIDA
5	CMTIN	ENTRADA
6	REMOTO +	SALIDA
7	REMOTO —	SALIDA
8	GND	—

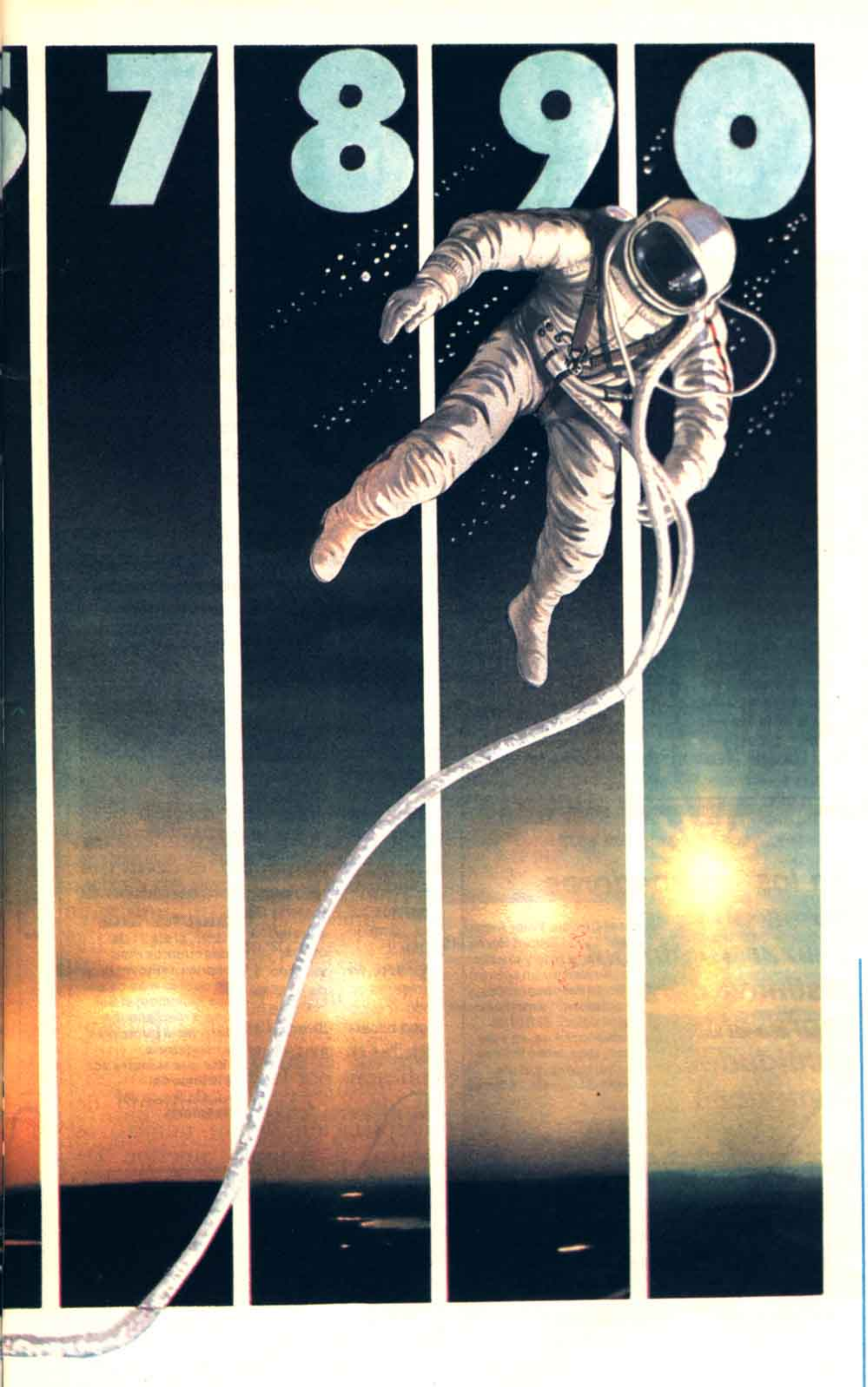
Figura: 8
Diagrama de las conexiones del port de E/S del cassette.

NOMBRE PIN	OBSERVACIONES
1. SALIDA DE VIDEO Y SEÑAL COMPUESTA 2. SEÑAL MODULADA RF	CONECTOR DIN 5 PINS O CONECTOR RCA 2 PINS CONECTOR RCA 2 PINS
CASSETTE	CONECTOR DIN 8 PINS
PORT E/S	CONECTOR AMP 9 PINS
IMPRESORA	CONECTOR 14 PINS
BUS DEL CARTUCHO	CONECTOR 50 PINS
SONIDO	CONECTOR RCA 2 PINS

Figura: 9
Los distintos conectores que pertenecen a una configuración completa de un sistema MSX.



Sistemas de numeración



Desde los tiempos más remotos, el hombre a utilizado sus manos para realizar una operación tan simple como es la de contar.

Curiosamente, la mayoría de los sistemas de enumeración antiguos se basaban en el número 10 (hecho normal, si consideramos que los árabes impusieron su cultura por aquellos tiempos), a excepción de los mayas e incas de América Central.

Como sabemos, el sistema decimal se basa en los valores posicionales de las cifras. Esto significa, que si queremos escribir 3, se entiende que son 3 unidades, pero si cambiamos su posición, podemos indicar 3 decenas (30) ó 3 centenas (300), etc. El número 3 adquiere un valor diferente según su posición, ésta tiene un valor, que es el del número (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) multiplicado tantas veces por 10 como indica su posición. Esto es lo mismo que hallar el valor de la base, 10, elevada a una potencia, que es la posición que ocupa el número. El primer valor de la potencia es 0, puesto que ésta es la de las unidades y no encontraremos ninguna potencia, excepto 0 (ya que cualquier número elevado a 0 es 1), que cumpla esta condición.

Sin embargo, en los ordenadores, no es posible utilizar diez símbolos distintos para representar cualquier número, ya que la máquina al funcionar con impulsos

eléctricos, sólo puede controlar dos estados, que son la presencia (1) o ausencia (0) de corriente. La información se recibe y almacena como lo que es, una combinación de 0 y 1. Estos se dispondrán dentro del ordenador de la forma que mejor le convenga a la máquina.

De esta manera, podemos entender cómo los ordenadores almacenan los datos en este sistema de numeración, ya que es el que permite una inmejorable representación de la disposición interna de la memoria.

Este sistema también es posicional, ya que el valor de los símbolos, en este caso el 1, depende del lugar que ocupe. Pero veamos cómo funciona el sistema posicional.

En la anotación decimal que utilizamos diariamente, todo número se representa mediante una sucesión de cifras 0, 1, 2, ... 9, donde el valor de una cifra está en función de la posición que ocupe dentro de un número, así 30 no es lo mismo que 300, ni que 3000.

Cuando escribimos un número en base 10 realizamos una operación dictada por la costumbre, aunque en realidad, utilizamos las cifras de que disponemos como multiplicadores de las potencias sucesivas de la base 10.

Las potencias de la base son los valores que se obtienen multiplicando entre sí tantos factores iguales a la base como indica el exponente. Por ejemplo:

10 0=1 potencia cero (unidades)
10 1=10 potencia uno (decenas)
10 2=10 × = 100 potencia dos (centenas)
10 3=10 × 10 × 10 = 1000 potencia tres (millares)

En un número decimal, cada cifra, empezando por la derecha y siguiendo por la izquierda, es un multiplicador de las potencias sucesivas de 10.

El riguroso procedimiento de la tabla siguiente permitirá asentar los conceptos vistos hasta ahora. El número 3605 se puede descomponer de la siguiente manera:

Número: 3 6 0 5
Potencias de la base: 10↑3 10↑2 10↑1 10↑0
Peso posicional: 1000 100 10 1
Valor (cifra × peso): 3000 600 0 5
Valor numérico: 3000 + 600 + 0 + 5 = 3605

Este ejemplo hará sonreír a más de uno, pero es básico a la hora de entender el concepto de las bases numéricas, ya sean binarias, octales o hexadecimales, que son las más utilizadas en los ordenadores de todos los niveles, desde los "mainframe" hasta los pequeños ordenadores personales.

En los ordenadores no se pueden utilizar 10 símbolos distintos para representar cantidades numéricas.

Comprendido el ejemplo será fácil entender que un número escrito en base 10 tiene su homólogo en cualquier otra base, siendo las más importantes las citadas anteriormente. Vamos a ver cuáles son los distintos símbolos utilizados en estas bases de numeración.

En sistema decimal, los símbolos son 10:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9
En sistema binario, los símbolos son 2:
0, 1
En base octal, los símbolos son 8:
0, 1, 2, 3, 4, 5, 6, 7
En base hexadecimal, los símbolos son 16:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Con estos símbolos, podremos proceder de la misma forma, a la hora de hacer cálculos, que con los símbolos decimales, sólo que existe una pequeña diferencia que la veremos cuando tratemos el tema de la operaciones con los sistemas numéricos.

De las bases 8 y 16 hablaremos más adelante. Empezaremos con el sistema binario, que es el más importante por el momento y a partir del cual se desarrollan las otras bases.

Sistema binario

Al igual que la notación decimal, el sistema de numeración binario es posicional, es decir, que un número es una combinación de 0 y 1 puestos en cualquier orden, donde el valor de ese número estará en función de la posición que ocupe el dígito 1.

En este caso, las cifras que ocupan las distintas posiciones se multiplicarán por las potencias sucesivas de 2. El procedimiento de interpretación de un número es similar al ejemplo anterior. De todos modos, otro ejemplo resultará interesante.

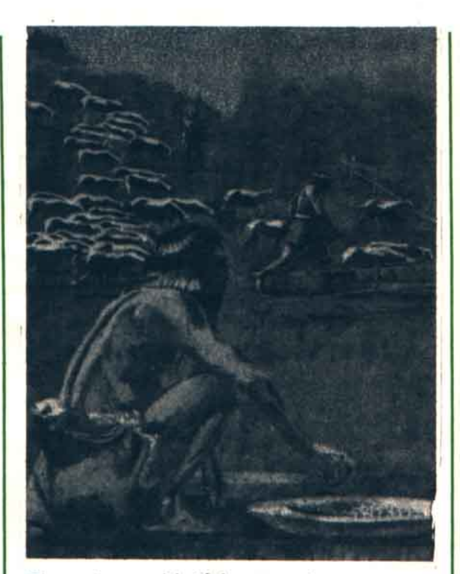
Número: 1 1 0 0
Potencia de la base: 2↑3 2↑2 2↑1 2↑0
Peso posicional: 8 4 2 1
Valor (cifra × peso): 8 4 0 0
Valor numérico: 8 + 4 + 0 + 0 =12

Por consiguiente, el valor 1100 en binario es igual a 12 en decimal. Hemos visto con este ejemplo que con cuatro dígitos binarios podemos representar el número 12. Pero si en lugar de haber colocado los "1" en esos lugares, hubiéramos realizado la operación de poner los cuatro dígitos a "1", ¿cuál sería su equivalente decimal? La solución debería hallarla el lector en base a

lo visto anteriormente, sin embargo, evitaremos que se repase la "lección". Colocando cuatro "1" obtendremos el valor de 15 en decimal. Este concepto va a resultar interesante en cuanto veamos el sistema de numeración hexadecimal, ya que este sistema se basa en 16 símbolos, del 0 al 15, que son todas las combinaciones de 0 y 1 que se pueden hacer en cuatro posiciones distintas. La realización de todas las combinaciones posibles lo dejaremos en manos de los lectores a los que daremos una pista, tiene que haber 16 combinaciones.

El sistema binario requiere gran cantidad de símbolos para indicar un valor numérico, por lo tanto resulta incómodo de usar a la hora de utilizar la máquina a fondo. Por este motivo se han introducido dos sistemas de numeración que permiten simplificar las operaciones a realizar por el ordenador. Estos sistemas son: el octal y el hexadecimal.

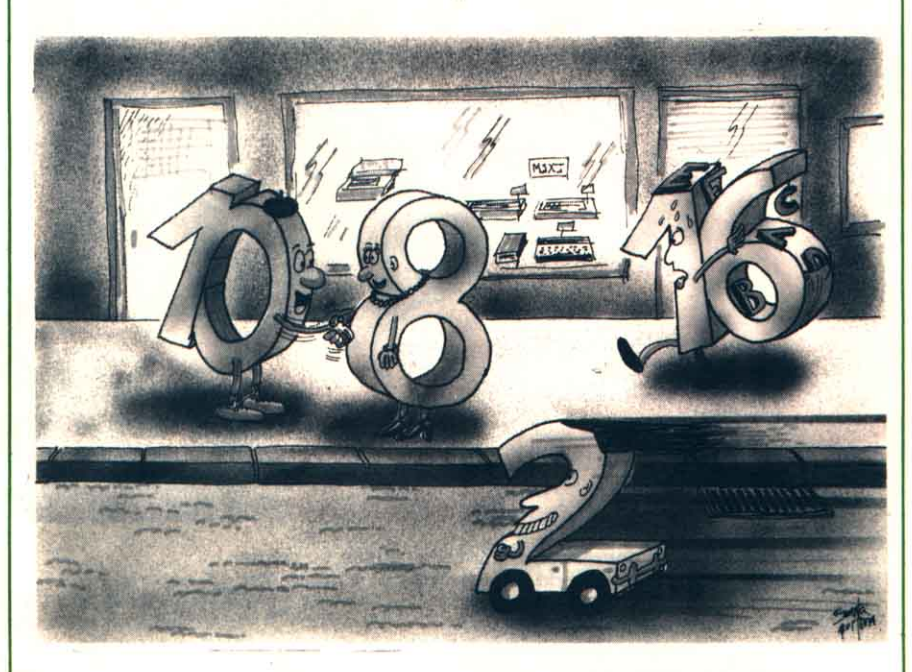
Estos sistemas permiten una representación compacta de los números, y por tanto, facilitan la escritura de éstos, sin tener que utilizar grandes cadenas de 0 y 1. Hay que tener en cuenta que el valor de un número es el mismo en binario, octal o hexadecimal, lo único que varía es la forma de expresarlo. Ya hemos visto como obtener un número decimal a partir de un número binario, veamos, pues, la operación inversa. La forma de proceder es contraria al ejemplo anterior. Allí vimos cómo se iban multiplicando los exponentes por el valor posicional, aquí haremos la división de tales números por la base a la que queremos efectuar el cambio. Por ejemplo, para obtener un número binario a partir de otro decimal, dividiremos sucesivamente entre 2 hasta lograr que el cociente sea 1. Con un ejemplo numérico se entenderá mejor los pasos a seguir. Tomemos el número 232 y hallemos su equivalente en binario.



El pastor prehistórico cuenta sus ovejas mediante dos vasijas con piedras. El número nacerá de la abstracción ideal de estas cuentas.

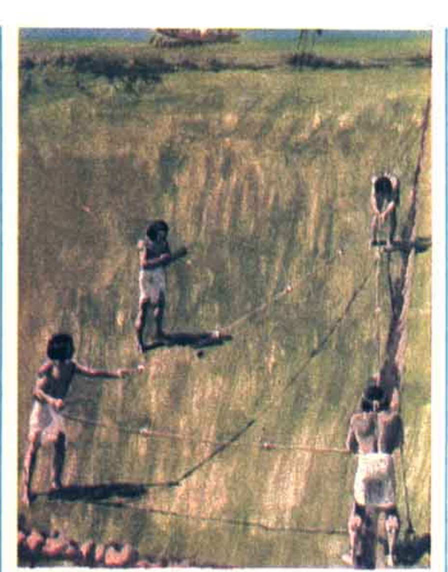
Decimal 232/2 = 116, resto = 0, potencia 2↑0
Decimal 116/2 = 58, resto = 0, potencia 2↑1
Decimal 58/2 = 14, resto = 1, potencia 2↑2
Decimal 29/2 = 14, resto = 1, potencia 2↑3
Decimal 14/2 = 7, resto = 0, potencia 2↑4
Decimal 7/2 = 3, resto = 1, potencia 2↑5
Decimal 3/2= 1, resto = 1, potencia 2↑6
Decimal 1 potencia 2↑7

Anotando los restos con el último cociente obtendremos el valor 11101000, que es el valor binario de 232 decimal. Para asegurarnos que el cambio de base ha sido correcto, tendremos que hacer la operación inversa, es decir, multiplicar los valores obtenidos por las potencias sucesivas de la base. Los cambios de base son simples de realizar una vez se tenga cierta soltura y muy útiles a la hora de manejar cualquier otro ordenador, ya que la mayoría de estos aparatos manejan estos sistemas de numeración.

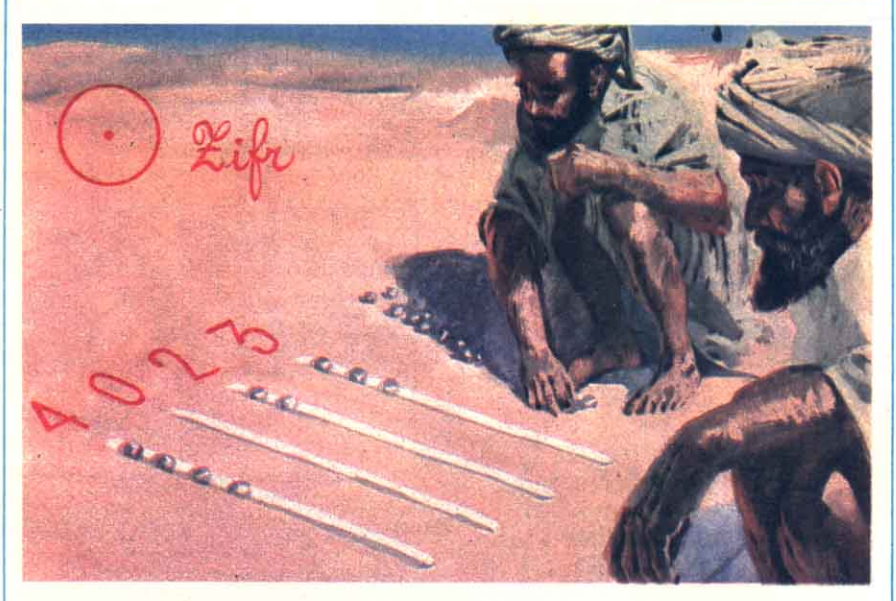


De todos modos, el BASIC MSX viene preparado con varias instrucciones para facilitar todo este trabajo. La instrucción &B (binario) representa un número binario, mientras que la expresión, BIN$ (expresión) calcula el valor binario de un número decimal.

Cada dígito de un número binario se denomina "bit" que viene



Los egipcios descubrieron la geometría al delimitar los confines de sus tierras de labor con cuerdas.



La numeración árabe, actualmente utilizada, posee su origen en la palabra cifra (cuyo sentido originario es el término Zifr = 0 cero.

del inglés *"binary digit"* (dígito binario), que como ya sabemos, sólo puede tener dos valores, 0 ó 1.

Es importante resaltar que 8 *bits* componen un *"byte"*, siendo este la longitd de una palabra de memoria.

Un *byte* puede representar un total de 256 valores, desde el 0000 0000 hasta el 1111 1111, pasando por todas las combinaciones intermedias.

Antes de pasar a otro sistema de numeración, veamos cómo utilizar las dos instrucciones que posee este BASIC vistas anteriormente. Para

El sistema de numeración binario es la base de todos los cálculos matemáticos y lógicos.

obtener un número decimal a partir de un número binario de 8 bits, realizaremos la siguiente rutina:

```
10 A% = &B 01101100
20 PRINT A%
RUN
```

En la línea 10 introduciremos el valor que deseemos obtener en decimal.

Para efectuar la operación contraria, haremos la siguiente rutina:

```
10 A% = 193
20 PRINT BIN$ (A%)
RUN
```

De esta forma obtendremos el valor binario del número situado en la instrucción 10.

Con estos dos ejemplos podremos realizar todas las conversiones que deseemos, sin tener que rompernos la cabeza.

Todos estos pasos son necesarios para que al lector le resulte más sencillo de entender y más fácil manejar las distintas bases numéricas.

A continuación explicaremos el siguiente sistema numérico en orden de importancia, el hexadecimal.

Sistema hexadecimal

Como ya hemos visto anteriormente, los ordenadores trabajan con números en base 16, lo cual facilita en gran medida la representación de números. Como ya vimos anteriormente, este sistema consta de 16 símbolos que, aunque sea repetitivo enunciar, son los siguientes:

Base decimal

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Base hexadecimal

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Con estos símbolos se logra una comodidad que no existía en los números binarios. Además, el pasar un número de esta base a la decimal es algo muy simple, como veremos a continuación.

Sabemos que un número en esta base se representa con cuatro bits,

Programas Sony para ordenadores MSX

A la orden.


Monkey Academy


Países del Mundo-1


Países del Mundo-2


Computador Adivino


Computer Billiards


The Snowman


Cubit


Character Collection


Stop the express (Para el Tren)


Hustler (Billar Americano)


Data cartridge


Quinielas y Reducciones


Home Writer


Sparkie


Aprendiendo Inglés-1


Binary Land


Creative Greetings


Aprendiendo Inglés-2


Antartic Adventure


Mastermind


Contabilidad Personal


Athletic Land


E.I.


Ficheros


El Ahorcado


Dorodon


La Pulga


Cosmos


Control de Stocks


Battle Cross


Mouser


Crazy Train


Ali baba


Juno First


Car Jamboree

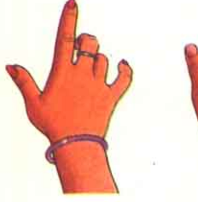

Tutor


Track and Field-1 (olimpiadas)


Blackjack


Track and Field-2 (olimpiadas)


Driller Tanks (Tanque Destructor)


Sonygraph


Ninja (El Samurai)


Les Flics



Y muchos más títulos

Ordenador Doméstico





Para lo que guste ordenar.



por lo que cualquier número binario que conste de 8 *bits*, lo podremos dividir en dos de cuatro y obtener así su valor hexadecimal. Antes hallábamos la combinación binaria del número decimal 232, hallemos ahora su valor en hexadecimal. Iremos paso a paso, para que se entienda mejor, pero antes de continuar, hemos de decir que este método, aunque rudimentario es de los más fáciles de aprender y de seguir, aunque existan otras formas de realizar el cambio de base.

PESO POSICIONAL	128	64	32	16	8	4	2	1
POTENCIA DE LA BASE	2↑7	2↑6	2↑5	2↑4	2↑3	2↑2	2↑1	2↑0
NUMERO BINARIO	1	1	1	0	1	0	0	0
VALOR(cifra x peso)	128+64+ 32 +0 +8 +0 +0 +0 = 232							

Esta era la representación a la que estamos acostumbrados, al haber realizado anteriormente un cambio de base, pero ahora que tenemos el número binario, dividámoslo en dos cifras de cuatro bits y tratémoslas independientemente.

PESO POSICIONAL	8	4	2	1	8	4	2	1
POTENCIA	2↑3	2↑2	2↑1	210	2↑3	2↑2	2↑1	2↑0
NUMERO BINARIO	1	1	1	0	1	0	0	0

A continuación hallemos el valor de las dos partes, también independientemente.

VALOR (valor x peso)	8+ 4 +2 + 0 = 14	8 + 0 +0 + 0 = 8

Una vez que tengamos el número binario descompuesto en dos cifras de 4 *bits* y obtenido el valor numérico de esas dos partes, iremos a nuestra tabla hexadecimal (donde tenemos los dieciséis símbolos que componen el sistema de numeración hexadecimal, del 0 a la F) y sustituiremos los valores de las dos partes por su correspondiente de la tabla. De esta forma tenemos que el 14 en decimal es la E en hexadecimal y que el 8 sigue

El sistema hexadecimal utiliza las letras A, B, C, D, E, F para representar los números 10, 11, 12, 13, 14 y 15.

siendo 8 en la obra base. En suma, el número hexadecimal equivalente al valor 232 decimal es E8.

Estos son los pasos para convertir un número decimal a hexadecimal, para realizar el proceso inverso, es decir, sustituiremos el valor hexadecimal por la combinación binaria o lo haremos mediante una fórmula que sirve, no sólo para esta base, sino para todas las que deseemos. El primer procedimiento ya lo conocemos, es el que hemos desarrollado a lo largo de este artículo, el segundo es el siguiente:

```
(decimal)=(primer dígito)* 16↑0 +
(segundo dígito)*16↑1 +(tercer
dígito)* 16↑2 +...
```

Este que parece más complicado, no lo es tanto. El número decimal, al pasarlo a otra base equivale a multiplicarlo por la base elevada a potencias sucesivas, por esta razón aparece como multiplicador el 16 (es la base de la que partimos, si partiéremos de otra base, la octal, por ejemplo, este número sería el 8). Haremos un ejemplo que aclarará este nuevo procedimiento.

Tomemos el número E8 y hallemos su valor decimal.

```
E8 = 8 * 16↑0 +E * 16↑1 = 8 *
1+ 14* 16 = 232
```

Sin embargo, todo este procedimiento se puede eliminar en el BASIC MSX, pues éste dispone de sendos comandos que realizan estas operaciones. La forma de convertir un número hexadecimal en su equivalente decimal es utilizando la función &H, mientras que para realizar la operación inversa se realiza con HEX$. Dos programas aclararán este paso:

```
10 X% = &F3DF
20 PRINT X%
RUN
```

Esta rutina halla el valor decimal del número hexadecimal de la línea 10, la siguiente rutina obtiene el equivalente hexadecimal del número decimal.

```
10 X% = 196
20 PRINT HEX$ (X%)
RUN
```

Estas rutinas son de suma utilidad a la hora de realizar programas en código máquina. Es interesante resaltar que la notación binaria se usa cuando realizamos dibujos o caracteres gráficos con *sprites*, ya que esta forma es la más sencilla a la hora de ver qué *bit* está puesto a 1 y cuál no. La notación hexadecimal, sin embargo, se deja para programar en código máquina puesto que es la mejor forma de asignar las direcciones de memoria Como vemos cada sistema de numeración tiene su utilidad en

cuanto al ordenador se refiere, cada uno tiene una función que le caracteriza, aunque es posible realizar sprites con valores hexadecimales y asignar direcciones de memoria con números binarios, pero es labor algo complicada, ¿cómo descomponer un número como &HBFFF en dígitos binarios? La pregunta es curiosa pero determinante para asentar las bases de que a cada sistema numérico le corresponde una función.

Sistema octal

Este sistema aunque sea el último en describir no es menos importante, no tiene tantas aplicaciones como los dos sistemas anteriores, pero es de suma utilidad puesto que alguna función puede ayudarnos a resolver. Esta base se caracteriza por utilizar solamente los 8 símbolos siguientes:

BASE DECIMAL:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9

BASE OCTAL
0, 1, 2, 3, 4, 5, 6, 7, 10, 11

Entrar en definir los procedimientos para convertir un número de esta base a otra sería redundar en la materia, por esta razón, sólo veremos con que instrucciones del BASIC MSX obtendremos los cambios que deseemos y rectificar la fórmula anterior que se vio acerca de los cambios de base de un número hexadecimal a su equivalente decimal, que también vale para este sistema, siempre y cuando se sustituya el valor 16 por el 8. De todos modos así queda la función:

(decimal) = (primer dígito) * 8↑0 + (segundo dígito) 8↑1 + (tercer dígito) * 8↑2 + . . .

Como vemos, es igual a la vista anteriormente, con la excepción del multiplicador es 8 en lugar de 16.

La base octal está a mitad de camino entre la hexadecimal y la binaria.

Al igual que en las dos bases anteriores, también existen dos rutinas que permiten el cambio de base, de octal a decimal y viceversa. Las rutinas son las siguientes:

```
10 X% = &05467
20 PRINT X%
RUN
```

Convertirá un número octal al sistema decimal, mientras que:

```
10 X% = 189
20 PRINT OCT$ (X%)
RUN
```

realizará la operación contraria, convertirá un número decimal al sistema octal.

De cualquier manera, la parte más interesante de los sistemas numéricos la veremos más adelante. El saber cómo opera el ordenador por dentro es de una ayuda difícil de estimar, hasta que uno no lleva bastante tiempo realizando programas. El MSX ofrece unas aptitudes en cuanto a los cálculos numéricos que harán las delicias de los expertos y de los que van camino de convertirse en ellos. No hay otro ordenador personal que admita trabajar con números en doble precisión con 14 dígitos significativos pero esto es materia que trataremos a su tiempo, ya que por el momento, basta saber que existen diversos sistemas de numeración y que los ordenadores trabajan en los sistemas binarios y hexadecimales. Dejaremos para futuras ocasiones las operaciones lógicas del Algebra de Boole, así como se efectúan sumas y restas en las distintas bases numéricas.

Para acabar, un pequeño cuadro de los 16 primeros números en las distintas bases.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Visto todo esto, muchos se preguntarán cómo se representan las fracciones de los números dentro de la memoria del ordenador.

Este es otro tema que trataremos en futuros artículos, ya que por el momento, interesa que los conceptos expuestos aquí no caigan en saco roto y que se entiendan, para de esa forma trabajar con más soltura. □



SVI-328 SISTEMA PROFESIONAL

El Sistema Profesional SVI tiene como núcleo el SVI-328, un ordenador de altas prestaciones cuya característica básica es su capacidad de ampliación. Con la adición de un SuperExpander —en cualquiera de sus versiones— se dispone ya del Sistema Operativo CP M y de su enorme biblioteca de programas que incluye procesadores de texto, hojas de cálculo, bases de datos, contabilidades, etc. La variedad de periféricos y expansiones disponibles hace del Sistema Profesional SVI-328 uno de los más completos y versátiles de cuantos existen.

- SuperExpander con uno o dos discos de 320 K, simple o doble cara.
- Tarjeta de 80 columnas.
- Tarjeta Centronics.
- Tarjeta Serie RS 232.
- Tarjetas de ampliación de memoria.

- Solicite a nuestros distribuidores la Oferta Especial "Value Pack".

P.V.P. Ordenador SVI-328...67.500 pts.

LA INFORMATICA DEL FUTURO

MSX SVI-728 PLUS

MSX es un standard universal para ordenadores personales que asegura la total compatibilidad tanto entre equipos como entre periféricos y programas. Definido y desarrollado por Microsoft, líder mundial en software, y por Spectravideo Internacional, ha sido ya adoptado por numerosos fabricantes de todo el mundo.

El ordenador personal SVI-728 PLUS añade a todas las ventajas de la norma MSX un diseño propio de un "profesional": 90 teclas (con "ñ", acentos, teclado numérico independiente...), 80 K de memoria RAM, sonido, gráficos de alta resolución, ranura y reset para cartuchos,...
... y una completísima gama de periféricos:

- Unidad de disco de 320 K (incluido sistema operativo CP M).
- Tarjeta de 80 columnas.
- Magnetófono
- Ampliación de memoria
- Modem
- Cable Centronics.
- Cartucho de conexión. Red Local.

P.V.P. 64.500 pts.



Los super-joysticks QUICKSHOT de precisión. Para SPECTRAVIDEO, AMSTRAD, ATARI, COMMODORE, etc.

MSX RED LOCAL DE COMUNICACIONES (LAN)

Hasta 32 ordenadores SVI-328, SVI-728 PLUS o cualquier otro del standard MSX con al menos 64 K de memoria RAM pueden conectarse como estaciones de trabajo a una estación "master" que controla la red. A una velocidad de transmisión de 230 Kbits por segundo y utilizando el Sistema Operativo CP M, un disco duro Winchester de 10 M almacena la biblioteca de programas comunes, los programas realizados por los distintos usuarios, el software de la Red, un "spooler" para la impresora, etc. Especialmente diseñada para su empleo en educación, la conexión del conjunto es extremadamente simple y de gran sencillez de operación.



SVI™
SPECTRAVIDEO

Avda. del Mediterraneo, 9
Tels. 433 45 48 - 433 48 76
28007 MADRID
Delegación en Cataluña:
Tarragona, 110. Tel. 325 10 58
08015 BARCELONA

DE VENTA EN El Corte Inglés Y TIENDAS ESPECIALIZADAS

Números romanos



Programar el ordenador para que realice cambios numéricos entre el sistema romano y árabe es algo que está al alcance de cualquier y más con la ayuda que os prestaremos, ya que pensamos seguir y explicar paso por paso la realización del programa.

Los ordenadores tienen la extraña "manía" de utilizar cualquier base numérica para realizar sus cálculos, ya sean en binario, octal, hexadecimal, decimal o hasta en coma flotante. Seguramente, si introducimos cualquier número romano, como por ejemplo, MCD, por mucha memoria que posea y por muchos bits que tenga el microprocesador, lo más que dirá será que "No entiendo". Máquinas menos pretenciosas a lo mejor se bloquean y no se obtienen resultados algunos. Pero por el momento, presentamos un programa que permitirá realizar semejantes cambios, de Arabe a Romano y viceversa. Además será gratificante ver como nuestro ordenador "aprender" a manejarse con este tipo de número. Después de todo, los números romanos se ven en todas partes, monumentos nacionales, en los índices de los libros, etc.

De cualquier manera vamos a tomar este programa como una lección de programación y, aunque duela decirlo, como la primera aproximación a las matemáticas.

El sistema numérico romano se basa en siete letras que son: M, D, C, L, X, V y I, que representan las cantidades de 1.000, 500, 100, 50, 10, 5 y 1, respectivamente. Podemos comprobar que cualquier número positivo tiene representación numérica en este sistema, que está en función de la posición que

ocupa cada letra dentro de un número.

En los siguientes sistemas numéricos, las cantidades se anotaban de izquierda a derecha, según la magnitud del número. El resultado final era la suma de los valores independientes de las letras.

El moderno sistema numérico romano, permite la combinación de dos letras cuyas magnitudes estén intercambiadas. En estos casos, la cifra más pequeña del par intercambiado, se resta en lugar de sumarlo. Por ejemplo: XXIV = 10 + 10 — 1 + 5 = 24.

Sólo se permiten intercambiar seis parejas que son: IV, IX, XL, XC, CD y CM, por cada par invertido, el segundo miembro ha de ser 5 ó 10 veces mayor que el primero. De esta forma, IM, no sería la representación válida del número 999, ya que M es 1.000 veces mayor que 1.

Exiten dos reglas que no permiten abusar de las magnitudes invertidas. La primera es que el número que precede al par sea mayor que el segundo miembro del par. Como resultado de aplicar esta regla, números como MDM y DDM no son permitidos. En segundo lugar, el número que precede el par ha de ser más pequeño que el primer miembro de la pareja. Por este motivo, un número como CDC no está permitido.

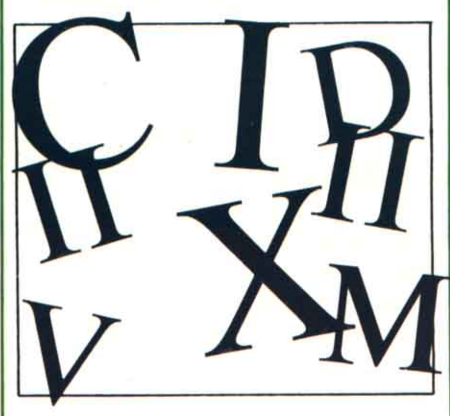
En el sistema numérico romano, las letras se pueden utilizar tres veces consecutivas. Por ejemplo, **XXXX no es una representación válida del número 40; en su lugar** se ha de utilizar la expresión XL. Como es normal, cada regla tiene su excepción y en este caso no iba a ser menos; la M se puede utilizar tantas veces como sea necesario.

Procedimiento de la conversión

Debido a que la situación de la letra es crítica dentro del contexto de un número al determinar su valor, comenzaremos explicando la **siguiente tabla, que muestra todas** las combinaciones posibles dentro de la secuencia de dos letras

	M	D	C	L	X	V	I
M	1000	500	100	50	10	5	1
D	0	0	100	50	10	5	1
C	800	300	100	50	10	5	1
L	0	0	0	0	10	5	1
X	0	0	80	30	10	5	1
V	0	0	0	0	0	0	1
I	0	0	0	0	8	3	1

Ahora aplicamos la tabla a una secuencia de dos números cualesquiera, que llamaremos a y b. Para encontrar el valor del número b precedido del número a, tendremos que observar la intersección de la fila a y columna b. En la intersección encontramos el número que sumaremos al valor final. Las combinaciones erróneas están señaladas con el 0.



Observar como la V después de una I suma 3 y no 5. Sin embargo, debido a que la letra I siempre suma 1, según la tabla, el efecto final de sumar 3 es el mismo que sumar 5 y restar 1. En otras palabras, IV = 1 + 3 = 4. La tabla también muestra este mismo efecto en varias combinaciones.

Pero veamos como se realiza una conversión más compleja, para la cual tomaremos el número DCIV. Cuando evaluamos la primera letra tendremos en cuenta que es la primera, por lo que no tendrá letra que le preceda. De manera que el valor inicial es el equivalente de esa letra, en este caso, 500.

Para obtener el valor de la segunda letra (C), buscaremos la intersección de la fila D (letra anterior) y columna C. En la tabla vemos un 100, que habrá que sumarlo al valor obtenido anteriormente, obteniendo un total parcial (puesto que todavía quedan por hallar dos valores) de 500 + 100 = 600.

El tercer valor se obtiene de la misma manera que el anterior. Nos posicionaremos en la intersección de la fila D con la columna I, sumaremos este valor (1) a la cantidad anterior (600) y obtendremos otro valor parcial que es 600 + 1 = 601. Con el tercer número procederemos de la misma forma que en las anteriores ocasiones. Posicionándonos en la intersección de la fila I con la columna V obtendremos un valor que sumado a la cifra anterior será el valor total de la transformación numérica, este es 3, que añadido a la cantidad anterior dará como resultado el valor decimal del número DCIV, siendo este 604.

La tabla en sí no muestra todas las combinaciones posibles de los pares de letras válidos. Hay que poner especial atención en la realización de estos pares, puesto que nos podemos encontrar en la desagradable situación de tropezarnos con una combinación que aunque parezca válida, no lo es tanto.

La operación de convertir números árabes a su equivalente romano es más sencilla. Empezaremos diciendo que todo número decimal, d, se puede expresar como la suma de diversos valores del sistema romano. La fórmula siguiente es aplicable en todos los casos.

d = a*M + b*CM + c*D + d*CD +e*C + f*XC + g*L + h*XL + i*X + j*IX + k*V + l*IV + m*I

donde las letras a, b, ..., m, representan números decimales positivos. **De esta forma para convertir números árabes a romanos, solamente tendremos que encontrar los factores de cada elemento (M, CM,**

```
5 REM Conversiones Numéricas
10 DIM T(7,7),L(7),F(13),FC$(13)
20 FOR TR = 1 TO 7
30 FOR TC = 1 TO 7
40 READ T(TR,TC)
50 NEXT TC
60 NEXT TR
70 DATA  1000,500,100,50,10,5,1
80 DATA  0,0,100,50,10,5,1
90 DATA  800,300,100,50,10,5,1
100 DATA 0,0,0,0,10,5,1
110 DATA 0,0,80,30,10,5,1
120 DATA 0,0,0,0,0,0,1
130 DATA 0,0,0,0,8,3,1
140 FOR TC = 1 TO 7
150 READ L(TC)
160 NEXT TC
170 DATA 4, 3, 3, 2, 2, 1, 1
180 FOR N = 1 TO 13
190 READ F(N)
200 NEXT N
210 FOR N = 1 TO 13
220 READ FC$(N)
230 NEXT N
240 DATA 1000,900,500,400,100,90,50,40,10,9,5,4,1
250 DATA M,CM,D,CD,C,XC,L,XL,X,IX,V,IV,I
260 C$="MDCLXVI"
270 CLS
280 PRINT "NUMEROS ROMANOS
290 PRINT
300 PRINT "Elige una opción:"
310 PRINT "  1- Número romano a árabe"
320 PRINT "  2- Número árabe a romano"
330 PRINT "  3- Salir del programa"
340 S=0
350 INPUT "Introduce 1, 2 o 3";S
360 IF S<1 OR S>3 THEN 290
370 IF S=3 THEN END
380 ON S GOSUB 620,400
390 GOTO 290
400 N$=""
410 PRINT
420 PRINT "Introduce el número o 'RETURN' para
430 PRINT "cancelar."
440 INPUT N$
450 IF N$="" THEN RETURN
460 N=VAL(N$)
470 IF N<=0 OR N<>INT(N) THEN 590
480 R$=""
490 FL=1
500 NT=N-F(FL)
510 IF NT<0 THEN 550
520 R$=R$+FC$(FL)
530 N=NT
540 GOTO 500
```

```
550 FL=FL+1
560 IF FL<=13 THEN 500
570 PRINT R$
580 GOTO 400
590 PRINT "Ese número no se puede convertir"
600 PRINT "Introduce números positivos"
610 GOTO 400
620 N$=""
630 PRINT
640 PRINT "Introduce el número o RETURN para"
650 PRINT "cancelar."
660 INPUT N$
670 IF N$="" THEN RETURN
680 TL=0
690 F=0
700 PL=4
710 PC=1
720 OC=1
730 D=1
740 RC=0
750 F$=MID$(N$,D,1)
760 CC=INSTR(1,C$,F$)
770 IF CC=0 THEN 1010
780 CL=L(CC)
790 IF CC<>PC THEN RC=1
800 IF CC=PC THEN RC=RC+1
810 IF RC>3 AND CC<>1 THEN 1040
820 IF F=1 AND CL>=PL THEN 1060
830 V=T(PC,CC)
840 IF V=0 THEN 1060
850 TL=TL+V
860 IF CC>=PC THEN 910
870 IF L(OC)<=PL THEN 1060
880 F=1
890 CL=L(PC)
900 GOTO 920
910 F=0
920 PL=CL
930 OC=PC
940 PC=CC
950 D=D+1
960 IF D>LEN(N$) THEN 980
970 GOTO 750
980 PRINT TL
990 GOTO 620
1000 PRINT "Carácter inválido: '";F$;"'"
1010 PRINT "Utiliza solamente (M,D,C,L,X,V,I)"
1020 GOTO 620
1030 PRINT "Demasiados ";F$;". El maximo es 3."
1040 GOTO 620
1050 PRINT "Secuencia de caracteres inválida"
1060 PRINT N$
1070 PRINT SPC(D-1);"^"
1080 GOTO620
```

etc). La mejor forma de entender los pasos es ilustrándolos con el siguiente ejemplo, donde vamos a transformar el número 3426.

Factores numéricos	Número romano acumulado
3426 - M = 2426	M
2426 - M = 1426	MM
1426 - M = 426	MMM
426 - CD = 26	MMMCD
26 - X = 16	MMMCDX
16 - X = 6	MMMCDXX
6 - V = 1	MMMCDXXV
1 - I = 0	MMMCDXXVI

Al final de la operación, tendremos el resultado de la conversión que nos indica 3426=MMMCDXXVI.

Observaciones sobre el programa Basic

Este programa está escrito en el BASIC de *Microsoft*, lo que quiere decir que funcionará en todos los ordenadores que posean esta versión del BASIC.

El primer bloque de líneas, de la 10 hasta la 260, preparan las distintas variables que se utilizarán a lo largo del programa. Estas son T (7,7) que es la tabla de la secuencia de los números romanos. L (7) almacena el orden de magnitud de cada número romano. El orden 4 indica los millares (M), orden 3 las centenas (D, C), orden 2 las decenas (L, X) y orden 1 las unidades (V, I). La variable F (13), almacena los trece factores de la conversión, vistos en el ejemplo anterior y FC$ (13) almacena el número romano correspondiente a cada factor.

El siguiente bloque está formado por las líneas que presentan el menú en la pantalla y son las líneas 270 a la 390.

Conversión árabe a romano

A continuación entramos una de las opciones de convesión. La primera que aparece es la de pasar números árabes a romanos. En la línea 470 se comprueba que el número introducido sea positivo, de ser así se ejecutarían las líneas 480 a 580 que son las que llevan a cabo la operación.

La variable R$ suma a su cadena un nuevo factor cada vez que restamos del número árabe, la letra correspondiente se añade a esa cadena. Por ejemplo, si FL = 2, entonces F(FL) = 500 y por consiguiente FC$(FL) = "D".

La línea 500 resta un factor del número que estamos convirtiendo, que es N, almacenando el resultado en la variable NT. Si NT es menos que 0, entonces el factor utilizado es demasiado grande y en la línea 550 se incrementará el índice para acceder al siguiente factor. Recordar que cuanto más grande sea el índice FL, menor será el valor del factor FC$(FL).

Este proceso se realizará hasta que los 13 factores se hayan probado. A estas alturas, R$ contendrá



el valor final en números romanos. La línea 570 imprimirá su valor y la 580 obligará al programa a ejecutar esta rutina nuevamente, que le pedirá que introduzca otro número árabe.

Los errores que podamos cometer al introducir el valor, se contemplan en las líneas 590 a 610, estas mostrarán un mensaje, para notificar el error y volverá a pedirle que introduzca otro número.

Conversión romano a árabe

Las líneas 620 a la 740, realizan la labor de convertir un número romano a decimal.

La variable TL almacena el total parcial del valor numérico de las letras. F indica si la letra leída anteriormente formaba parte de un par invertido o no.

La variable PL es el orden de magnitud de la letra anterior y PC es el número de la columna de esa letra, siendo M la columna 1, D la columna 2, etc. Las variables CC y CL de las líneas 760 y 780 almacenan la información correspondiente sobre la letra que se está tratando en ese momento. Al evaluar la primera letra de esta combinación, es obvio que no existirá algún valor previo, correspondiente a las variables PC y OC. Sin embargo, como la estructura del programa está dispuesta para evaluar estas dos variables, actuaremos como si las dos letras leídas anteriormente fueran M.

No podía faltar un puntero, este controla la letra que estamos utilizando en ese momento.

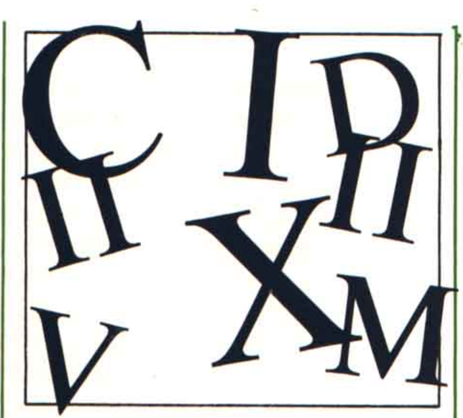
Para comprobar que la letras utilizadas no sobrepasen el límite de 3, existe un contador encargado de realizar esa función, este es RC. Si su contenido excede del límite, 3 en este caso, el número romano será inválido (excepción hecha con la letra M). Las líneas 760 y 770 aseguran que la letra sea una de las 7 permitidas, mientras que la línea 780 almacena el orden de la magnitud de la cifra que contiene CL.

El bloque formado por las líneas 790 a 810, evitan que aparezcan más de 3 letras iguales dentro del valor de un número, mientras que la línea siguiente (820) refuerza lo que hemos dicho anteriormente acerca de que el número siguiente al par ha de ser menor que el primer miembro de este.

F = 1 indica que la letra anterior forma parte del número invertido. En este caso, la primera letra tendrá que ser de menor magnitud que el primer miembro del par. Las líneas 830 a 850 hacen referencia de la tabla indicada al principio, en la que se determina el resultado de la conversión de PC a CC. De este bloque destacaremos la

línea 840, encargada de detectar las secuencias inválidas para, en caso de que ocurra, salte a otra rutina (línea 1050) que maneja los errores. Si la secuencia es correcta, la línea 850, se encargará de sumar los valores de los números romanos, almacenados en la variable V, al total final, cuya variable es TL.

El comprobar si la combinación de pares invertidos es la correcta es la función del bloque formado por las líneas 860 a 940. Si el valor de la cifra que estamos convirtiendo, es menor o igual que la letra anterior, entonces no tenemos pares invertidos. La comparación de CC >= PC realiza esta función. Si CC < PC, entonces tenemos pares invertidos. Las líneas 870 a 900 comprueban las secuencias inválidas, como puede ser DDM. Las varia-



bles que actualizan la magnitud del carácter anterior PL, el carácter antiguo OC y el carácter anterior PC, son las correspondientes a las líneas 920 a 940. No confundir el valor de la magnitud del carácter, con el carácter en sí.

Finalizando, tenemos el bloque formado por las líneas 950 a 990 que continuan el proceso hasta que no queda número alguno que convertir, y la rutina que detecta los valores erróneos de las cantidades introducidas en la opción de los números romanos, estas líneas (1000 a la 1080) imprimen el mensaje correspondiente.

Ejecutado el programa, es fácil seguirle los pasos, especialmente si mientras tecleaba las instrucciones leía esta explicación. Para comprobar el correcto funcionamiento del programa, lo mejor es introducir un número en cualquiera de las dos opciones y una vez obtenido el resultado, introducirlo con la opción inversa. Si el resultado es el mismo, el programa está bien, de lo contrario habrá que repasar línea a línea hasta encontrar el error.

MSX-DOS, sistema



Que un ordenador personal posea características típicas de los *mainframe* no es extraño, al fin y al cabo, muchos fabricantes quieren convertir sus pequeñas máquinas en el segundo equipo de ordenadores de más talla y MSX, no iba a ser menos. Para ello ha creído conveniente introducir en sus ordenadores un sistema operativo que le haga compatible, no sólo con el resto de los ordenadores MSX, sino con los grandes ordenadores que tengan este sistema operativo, de esa forma, el que tenga en su oficina un ordenador grande con un MS-DOS, puede llegar a casa y "conectarse" al "pequeño" y continuar con la tranquilidad propia de saber que se está en casa.

El **DOS** del **MSX** se anunció en octubre de 1983, como sistema operativo para microprocesadores de 8 bits. ¿Qué es un sistema operativo? Se puede decir, que es un programa o una colección de programas que permiten un aprovechamiento máximo de las características del ordenador. Tiene múltiples fun-

operativo de verdad

ciones, entre las que podemos citar la de controlar la entrada y salida de información o llevar a cabo la conexión con los distintos periféricos, ya que, cuando nosotros conectamos cualquier aparato al ordenador, no observamos nada extraño, pero si nos limitamos a estudiar el comportamiento interno de la máquina, veremos como es detectado este nuevo periférico añadido o como reparte el ordenador nuestro programa por la memoria del aparato. En suma, este conjunto de programas permite aprovechar al máximo las posibilidades del ordenador.

En este artículo, no pretendemos profundizar en las características de este sistema operativo, sino acercar más al usuario del **MSX** a una característica que sólo poseen los ordenadores grandes. Normalmente, la situación física del sistema operativo, no es siempre la misma. Por situación física, queremos decir, que puede ocupar cualquier parte de la memoria, no un lugar específico. Esta se puede dividir en una o dos partes, que pueden ser residente en el sistema (cuando está incorporado dentro de un chip, de forma que cuando encendemos el ordenador, automáticamente se pone en marcha) o se puede cargar a partir de disco. En el primer caso, estará almacenada en memoria *ROM (Read Only Memory*-Memoria Solo Lectura), de manera que nunca se pierda y en el segundo caso, tendremos que cargar del disco el sistema operativo cada vez que conectemos el aparato.

En cualquiera de los dos casos, lo único que nos interesa por el momento, es el juego de instrucciones y el efecto que tienen sobre nosotros.

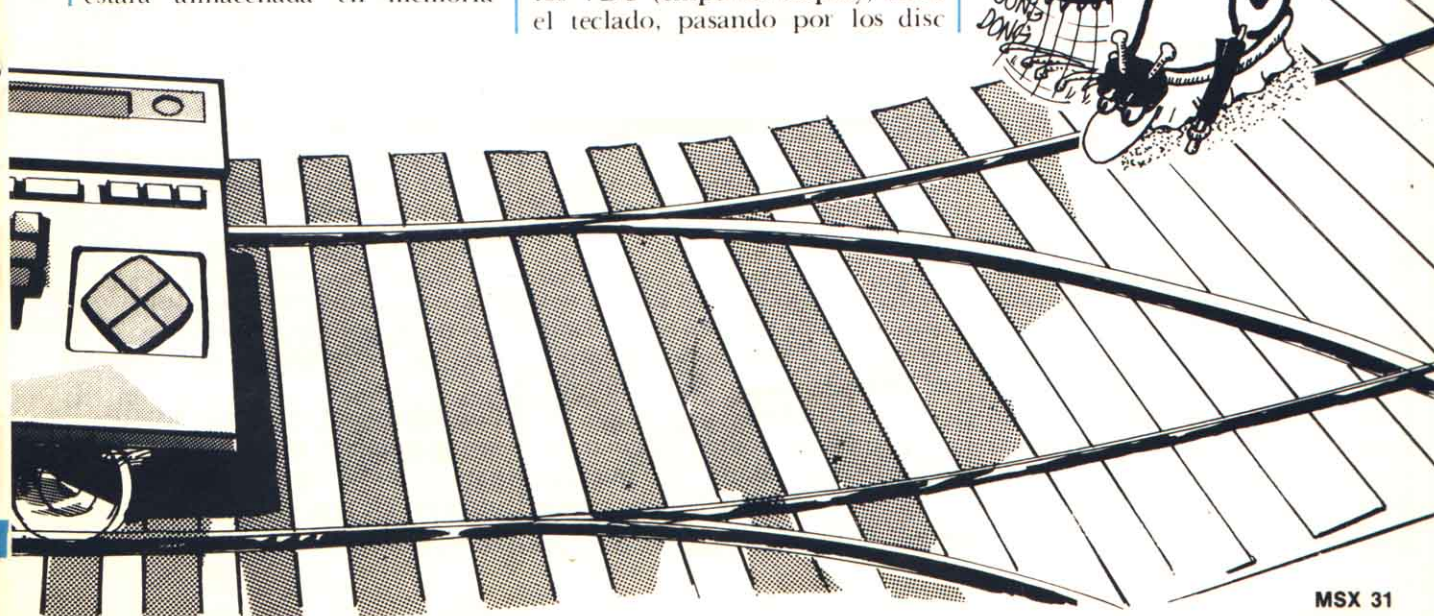
Para hacernos una idea de la importancia que tiene el DOS dentro del contexto de los ordenadores, pensemos en la función que realiza un controlador aéreo en un aeropuerto. Estos son los encargados de optimizar y conjuntar la entrada y salida de aviones, procurando no retrasar los vuelos ni crear colas de espera. Para ello disponen de todos los elementos necesarios para estar informados al segundo de lo que ocurre con los vuelos, ya sean nacionales o internacionales.

El **MSX** no es un aeropuerto, aunque poco le falta, pero ha de estar informado de todo lo que ocurre en su interior, pues nosotros seremos los beneficiados o perjudicados.

Entonces ¿cuál es su función? Tomando al **MSX** como ejemplo, observaremos que el **DOS** es el corazón que controla ciertas partes del sistema. Estas pueden ser desde los **VDU (chips del display)** hasta el teclado, pasando por los disc drives o por las impresoras. Suministra la información necesaria a estos aparatos para que cada uno realice su función a su debido tiempo. Por ejemplo, cuando cargamos un programa, el ordenador prepara la memoria **RAM** con instrucciones para que ese programa se pueda repatir en las diversas zonas de la memoria, organizando y ejecutandolo según las instrucciones. De la misma forma, existen comandos e instrucciones que controlan los dics drives, el acceso a ellos y la distribución de la información dento de las pistas.

COMPATIBILIDADES

Las funciones que hemos relatado anteriormente son parte de las "obligaciones" de un sistema operativo. Pero estos no son, en la mayoría de los casos, compatibles ni universales. Para que existiera una compatibilidad entre sistemas operativos, ya sea en ordenadores personales o en **"mainframes"**, se crearon el **CP/M** y el **MS-DOS**, cuyo objetivo era la de diseñar y ejecutar programas para ordenado-





res de distintas marcas. De cualquier manera y en algunos casos, encontraremos algún "ingeniero" dispueto a retocar el **CP/M** adquirido en una tienda para que pueda ejecutarlo en su ordenador. Entre las múltiples funciones que realiza el **DOS**, podemos destacar el acceso a cualquier fichero, ordenarlo, intercambiarlo, pero a lo mejor no podrá ser ejecutado y aunque se anuncia la existencia de un estándar, dentro de los sistemas operativos, todavía no está garantizada.

Para poder entender correctamente la función del **DOS**, veamos primeramente, como funciona el **MS-DOS**, padre del **MSX-DOS**.

El **MS-DOS** lo diseñó **Microsoft**, del cuál también partió la idea del **MSX** y consiste de una serie de programas. Este sistema operativo fue diseñado con el **IBM-PC** in mente y consta de cuatro partes o paquetes principales, preparados especialmente para ejecutar en cualquier sistema con disco, ya sea flopy o Winchester. Estas partes son:

BOOT- Este programa permite la autoejecución del sistema, cargando el **MS-DOS** desde el disco.

DOS.COM- Programa clave, puesto que es el organizador de todo el sistema de disco.

BIO.COM- Prepara una zona física de trabajo para un determinado diskette.

COMMAND.COM- Interpreta y procesa las instrucciones de entrada y salida.

Este último es el responsable de "traducir" los comandos introducidos por el usuario, que pueden ser internos o externos. Las instruse internos o externod. Las instrucciones internas están incorporadas al sistema y permiten borrar, cambiar de nombre, copiar ficheros, etc. mientras que las instrucciones externas se pueden cargar desde disco. Estos comandos se utilizan para cargar programas, introducir la fecha, formatear discos, etc. además de todas las utilidades añadidas por el usuario.

FORMATO DE DISCOS

Toda información, sean datos o programas, grabados en disco reciben la denominación de fichero. Estos se almacenan con un nombre que el usuario les da y tienen la posibilidad de poderse borrar o alterar, gracias a la colaboración del **DOS**.

El sistema operativo en disco (**DOS-Disk Operating System)**, es, como podemos comprobar, el centro neurálgico de un ordenador, de cualquier ordenador, y en la mayoría de los casos, compatibles con otros ordenadores con el mismo **DOS**. La compatibilidad absoluta todavía es casi imposible, parte de la culpa la tienen los fabricantes de diskettes, cuyos formatos no lo permiten. Los formatos de los discos se refieren a la estructura interna del soporte. Si nos paseamos por el mercado de los diskettes, podremos comprobar que existen cientos de marcas, con cientos de formatos, poniendo punto final a la compatibilidad de un equipo.

Un sistema operativo es una colección de programas que permiten aprovechar al máximo las características del ordenador.

El problema más complicado estaba resuelto, que los fabricantes se pusieran de acuerdo en preparar un estándar, pero no así la forma de almacenamiento de la información. Pero no todo está perdido, los programas se podrán ejecutar en diversos ordenadores, sólo que a la hora de grabarlos habrá que configurar el programa.

Aunque para hablar sobre los formatos de estos discos, hay que conocer la organización interna del dispositivo. Este tiene una serie de pistas que pueden ser 40 u 80, cada una de las cuales está dividida en sectores. Un formato muy típico de un diskette es que posea 40 pistas con 16 sectores por pista, la información se graba en 512 bytes por sector en formato de doble densidad, donde la primera pista de un disco (pista 0) contiene toda la información posible acerca de su contenido, siendo un punto de referencia para el sistema. El sector 1 de la pista 1 contiene la información del registro que arranca el sistema y los dos sectores siguientes contienen la tabla de dirección de los ficheros. Esta tabla indica al **DOS** si un sector se utiliza o no, es un puntero que marca la dirección del siguiente sector. En los sectores 4 a 7 de la pista 0, encontraremos el directorio, que es un índice de los ficheros y contiene la dirección de los bloques de datos existentes en el diskette. Sin embargo, el usuario no tiene porque preocuparse de toda esta información, que es misió
sión del sistema operativo. Este debe organizar la información física para que de esa forma, sea el usuario el que se preocupe de preparar la información lógica.



Esto que acabamos de describir, es el formato de un diskette de un ordenador **IBM**, algo distinto al **MSX-DOS** que utilizan estos ordenadores. Los comandos usados por el sistema operativo son distintos aunque tengan el mismo nombre. Por ejemplo, para borrar un fichero de un diskette, el usuario tendrá que teclear instrucciones como **DELETE** o **ERASE**. Teniendo conocimientos de inglés, podrá dialogar con el ordenador, al tener las instrucciones el mismo nombre que la función que ha de realizar. Habiendo hecho todos estos comentarios acerca del **MS-DOS**, entonces nos preguntaremos, ¿dónde situamos el **MSX-DOS**? ¿es estándar? La información acerca del **MSX-DOS**, está ahí y todo lo que se pueda añadir está en función de **Microsoft** y de los fabricantes del **MSX**, aunque todas las bazas no se han jugado todavía.

La idea del **MSX-DOS**, es un ambicioso plan que agrupa a todos estos ordenadores, que consiste en la posibilidad de que todos funcionen con el **DOS**, basado en el formato **MS-DOS** y que permitirá el intercambio de la información entre procesadores de 8 y 16 bits, convirtiendo, así el estándar **MSX** en un compatible de **"mainframes"** que tengan un sistema operativo **MS-DOS** .Esto permitirá un acceso a todos los posibles lenguajes de programación de **Microsoft.**

Si recorriéramos el catálogo del usuario del **MSX-DOS** tendríamos la impresión de estar hojeando el catálogo del **MS-DOS**. Además iniciarse en esta configuración es igual para todos los equipos, introduciremos la fecha, al igual que en el **MS-DOS**, y se nos preguntará por los *drives* A B cuando llegue el caso. Por más que veamos características de este sistema operativo, no hay que pensar nunca de que estamos trabajando con el **MS-DOS.**

Es muy fácil pensar que los distintos diskettes de diversos fabricantes puedan ser compatibles, pero hasta el momento, el único estándar que se asume, es el diskette de 3.5 pulgadas de Sony, aunque fabricantes como Hitachi, tienen puestas sus miras en su diskette, que claro está es de distinto tamaño que el anterior y tiene sus propias características.

El DOS es para el MSX como los controladores aéreos lo son para nosotros, elementos indispensables para una buena circulación.

En realidad, con la variedad de tamaños existentes, desde el 3.5 hasta el de 8 pulgadas, parece difícil una coordinación para obtener una medida en común. Por lo que se ve, cada fabricante se dedicará a ofrecer su producto. Esto motivó que Spectravideo lanzara al mercado un diskette de 5.25 pulgadas, mientras Sony seguirá con el de 3.5. **Microsof**, por su parte, no pone reparos a la hora de elegir el tamaño idóneo, esto es el punto negativo de esta estandarización, que obliga al usuario a buscar las distintas posibilidades dentro de un mercado que cada vez se está ampliando más. La respuesta a semejante problema está en manos de los fabricantes. Tendremos que esperar y ver que dirección van a tomar los acontecimientos.

¿Qué otras posibilidades ofrece el **MSX**? La guía del usuario expone los siguientes comando, a los que dedicaremos un artículo el mes que viene.

BLOAD	LOAD
BSAVE	LSET
CLOSE	RSET
COPY	MERGE
DSKO	NAME
FIELD	OPEN
FILES	PRINT #
FORMAT	USING
GET	PUT
KILL	RUN
LINE	SAVE
INPUT #	SYSTEM

La mayoría de estos comandos son permitidos en casi todos los ordenadores, unificando más la diversificación de criterios. Además existen unos parámetros a seguir por los fabricantes de software que han de cumplir. Con estos parámetros y datos técnicos los más experimentados podrán modificar algunas características del **MS-DOS**.

Otros puntos que todavía no están claros hacen referencia a la cantidad de ficheros que se pueden almacenar en un diskette.

De lo estamos seguros, es de que las especificaciones acerca de la compatibilidad de los diskettes no están nada clara. De cualquier manera, el **MSX-DOS** están empezando a crecer y hasta que llegue a su mejor momento tendrán que pasar algún tiempo, estaremos pendientes de los fabricantes y auqnue estos se dediquen a fabricar su diskette tendremos sistema operativo para rato, por lo menos el **DOS** seguirá siendo la mejor opción. □



Dados

Como ¿qué no habeís traído los dados? Si alguna vez has tenido este problema ya teneis solución. Conecta el MSX y ejecuta este sencillo programa que hará las veces de dado. Este aparecerá una vez por cada lado de la pantalla al pulsar la barra espaciadora.

El dado está formado por sprites de 16 x 16 pixels normales, de ello se encarga la instrucción **SCREEN** de la línea 20. A su vez, en ella definimos el color del sprites así como el del fondo.

El programa no tiene más función que mostrar la creación y el movimiento de un gráfico generado por este *BASIC*. Sería muy útil para el lector, retocar un poco el programa para que en lugar de mostrar un dado, fueran dos, añadiendo además algo más al sonido que tiene.

```
10 REM **DADOS**
20 CLS:SCREEN 2,2:COLOR 11,1,1
30 OPEN "grp:" FOR OUTPUT AS #1
40 PRESET(90,10):PRINT #1,"* DADOS *"
50 CLOSE #1:KEY OFF
60 DIM A$(4),X(16),Y(16)
70 D$=STRING$(32,255)
80 MID$(D$,1,1)=CHR$(127)
90 MID$(D$,16,1)=CHR$(127)
100 MID$(D$,17,1)=CHR$(254)
110 MID$(D$,32,1)=CHR$(254)
120 SPRITE$(6)=D$
130 FOR I=0 TO 4:FOR J=1 TO 5:READ D$
140 A$(I)=A$(I)+CHR$(VAL("&h"+D$))
150 NEXT:NEXT
160 FOR I=0 TO 5:D$=CHR$(0)
170 FOR J=0 TO 5:READ D
180 IF J=3 THEN D$=D$+CHR$(0)
190 D$=D$+A$(D)
200 NEXT:SPRITE$(I)=D$
210 NEXT
220 FOR I=0 TO 8 STEP .5
230 X(I*2)=I*25
240 Y(I*2)=5*I*I-30*I+75
250 NEXT
260 S=0:D=1
270 A=50
280 FOR I=0 TO 16
290 N=INT(RND(-TIME)*6)
300 IF N=0 THEN C=8 ELSE C=1
310 A$="l16t250n"+STR$(A):A=A-1
320 PLAY A$
330 PUT SPRITE 0,(S+D*X(I),Y(I)),C,N
340 PUT SPRITE 1,(S+D*X(I),Y(I)),15,6
350 FOR J=0 TO 36:NEXT
360 NEXT
370 D$=INKEY$:IF D$="" THEN 370
380 D=-1*D:S=S XOR 255:GOTO 270
390 REM Generación del dado
400 DATA 0,0,0,0,0,1,3,3,1,0
410 DATA 80,c0,c0,80,0,30,78
420 DATA 78,30,0,c,1e,1e,c,0
430 DATA 0,1,0,0,2,0,3,0,0,0,0,4
440 DATA 3,1,0,0,2,4,3,0,3,4,0,4
450 DATA 3,1,3,4,2,4,3,3,3,4,4,4
```

Coches locos



LOS JUEGOS ELECTRONICOS

Correr dentro de un circuíto es algo que no se hace todos los días, sin embargo, con un MSX, no sólo correrá el coche sino que también lo hará la imaginación.

Con este programa comprobarás tus reflejos, esquivando el coche que te preseguirá por todos los rincones del circuito hasta hacerte estrellar. De paso, mientras corres por el circuíto, dedícate a comerte los puntos que aparezcan a lo largo del camino. Veamos la realización de este programa.

El programa está estructurado en 6 bloques cuyas misiones son las siguientes. Las líneas 10 a 340 son las encargadas de inicializar la pantalla, dibujar el circuíto y pone los contadores de los coches que nos quedan así como el record del momento.

Seguidamente, comienza el bloque formado por las líneas 350 hasta la 1040, donde se controla el coche y se comprueba si hemos recorrido el circuíto entero.

Las líneas 1050 hasta la 1290 se ejecutan cuando ha finalizado el recorrido con éxito y le dan la opción de jugar nuevamente. El bloque siguiente, formado por las líneas 1300 a la 1610, son las encargadas de crear los coches orientados hacia los cuatro puntos cardinales. Por último, queda la parte más elemental, la creación del circuíto. Este se realiza mediante sentencias datas, donde están los diversos valores de todas las coordenadas a tener en cuenta.

```
10 REM Coches Locos
20 REM Inicializacion
30 SCREEN 2,2: COLOR 15,1,1
40 KEY OFF: WIDTH 30: CLS
50 CLEAR 400,&HDFFF
60 DIM C(23,22)
70 SC=0: HS=0: ST=0
80 ON SPRITE GOSUB 1180
90 GOSUB 1300
100 OPEN "grp:" AS #1
110 DRAW "bm60,60"
120 PRINT #1,"Un momento..."
130 CLOSE #1
140 GOSUB 1620
150 CA=3 : FU=1000
160 IF SC>HS THEN HS=SC
170 SC=0
180 X=21: Y=11: F=3
190 TX=1: TY=1: TF=1
200 OPEN "grp:" AS #1
210 DRAW "bm198,25"
220 PRINT#1,"Coches"
230 DRAW "bm208,45"
240 PRINT #1,"Locos"
250 LINE (220,120)-(250,130),1,BF
260 DRAW "bm195,120"
270 PRINT #1,"coches";CA
280 DRAW "bm195,150"
290 PRINT #1,"Record"
300 DRAW "bm195,165"
310 PRINT #1,"  ";HS
320 CLOSE #1
330 PUT SPRITE 1,(236,89),4,3
340 PUT SPRITE 2,(13,9),7,1
350 K$=INKEY$:IF K$="" THEN 350
```

```
360 FOR J=228 TO 171 STEP -8
370 FOR I=1 TO 50:NEXT I
380 PUT SPRITE 1,(J,89),4,3
390 NEXT J
400 SPRITE ON
410 KK=STICK(ST)
420 IF C(X,Y)>0 THEN 470
430 C(X,Y)=1: SC=SC+10: FG=FG+1
440 SX=X*8+10: SY=Y*8+10
450 LINE(SX,SY)-(SX+2,SY+2),1,BF
460 IF FG=FX THEN 1060
470 IF KK=0 THEN 630
480 IF KK=3 THEN RX=X+1:RY=Y:RF=1
490 IF KK=5 THEN RY=Y+1:RX=X:RF=2
500 IF KK=7 THEN RX=X-1:RY=Y:RF=3
510 IF KK=1 THEN RY=Y-1:RX=X:RF=4
520 D=C(RX,RY)
530 RRF=RF+2:IF RRF>4 THEN RRF=RRF-4
540 IF F=RF OR F=RRF THEN 630
550 IF C(X,Y)<>2 THEN 620
560 IF D=9 THEN 630
570 IF RF=1 THEN RX=X+2
580 IF RF=2 THEN RY=Y+2
590 IF RF=3 THEN RX=X-2
600 IF RF=4 THEN RY=Y-2
610 GOTO 730
620 IF D<>9 THEN F=RF: GOTO 730
630 IF F=1 THEN RX=X+1:RY=Y
640 IF F=2 THEN RY=Y+1:RX=X
650 IF F=3 THEN RX=X-1:RY=Y
660 IF F=4 THEN RY=Y-1:RX=X
670 D=C(RX,RY)
680 IF D<9 THEN 730
690 FU=FU-10
700 F=F+1
710 IF F>4 THEN F=F-4
720 GOTO 810
730 IF D<4 OR D>6 THEN 790
740 ON D-3 GOTO 750,770,780
750 Y=13:IF RND(1)<.5 THEN Y=9
760 X=15:F=3:GOTO 800
770 Y=13:GOTO 760
780 Y= 9:GOTO 760
790 X=RX:Y=RY
800 FU=FU-1
810 PUT SPRITE 1,(X*8+4,Y*8+1),4,F
820 IF C(TX,TY)<>3 THEN 860
830 TF=TF-1: IF RND(1)<.5 THEN TF=TF+2
840 IF TF<1 THEN TF=TF+4
850 IF TF>4 THEN TF=TF-4
860 IF TF=1 THEN XRX=TX+1:YRY=TY
870 IF TF=2 THEN YRY=TY+1:XRX=TX
880 IF TF=3 THEN XRX=TX-1:YRY=TY
```

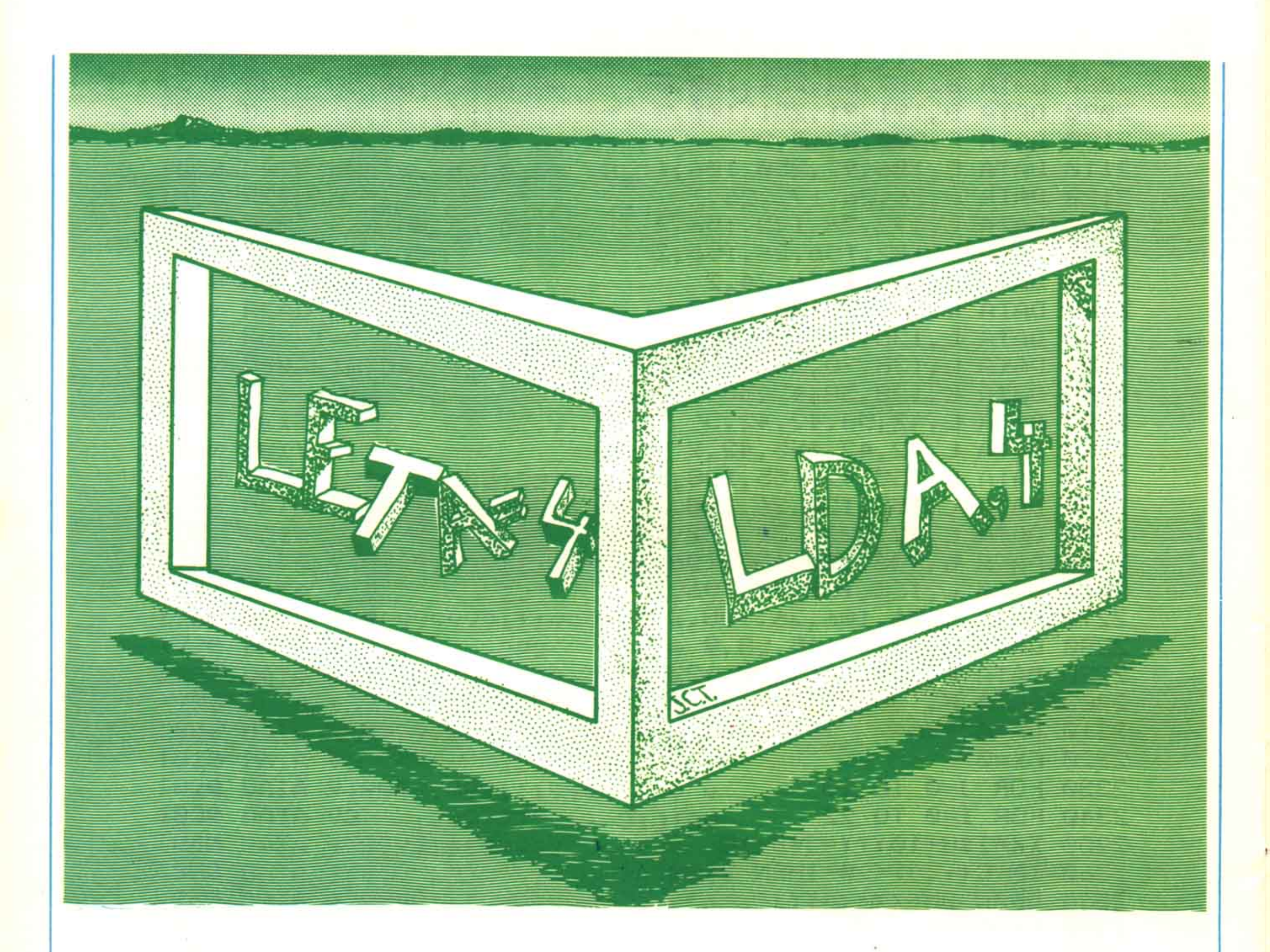
```
890 IF TF=4 THEN YRY=TY-1:XRX=TX
900 D=C(XRX,YRY)
910 IF D<9 THEN 960
920 TF=TF-1: IF RND(1)<.5 THEN TF=TF+2
930 IF TF<1 THEN TF=TF+4
940 IF TF>4 THEN TF=TF-4
950 GOTO 1030
960 IF D<4 OR D>6 THEN 1020
970 ON D-3 GOTO 980,1000,1010
980 TY=13: IF RND(1)<.5 THEN TY=9
990 TX=15: TF=3: GOTO 1030
1000 TY=13: GOTO 990
1010 TY=9: GOTO 990
1020 TX=XRX: TY=YRY
1030 PUT SPRITE 2,(TX*8+4,TY*8+1),7,TF
1040 GOTO 410
1050 LINE(40,60)-(150,140),1,BF
1060 DRAW"BM40,70"
1070 OPEN"GRP:" AS #1
1080 PRINT #1," ENHORABUENA!!"
1090 SC=SC+FU
1100 DRAW"BM50,85"
1110 PRINT #1,"Puntos=";SC
1120 DRAW"BM35,120"
1130 PRINT #1,"  Otra partida?"
1140 CLOSE #1
1150 K$=INKEY$: IF K$="n" THEN END
1160 IF K$="s" THEN CLS:GOTO 100
1170 GOTO 1150
1180 SPRITE OFF
1190 FOR J=1 TO 10
1200 FOR I=1 TO 4
1210 PUT SPRITE 1,(X*8+4,Y*8+4),J+I,I
1220 FOR K=1 TO 10: NEXT K
1230 NEXT I
1240 NEXT J
1250 CA=CA-1: IF CA>-1 THEN 1290
1260 OPEN"GRP:" AS #1
1270 LINE (40,60)-(150,140),1,BF
1280 GOTO 1090
1290 RETURN 180
1300 REM Sprites
1310 RESTORE 1420
1320 FOR J=1 TO 4
1330 SP$=""
1340 FOR I=1 TO 32
1350 READ D$
1360 D$=CHR$(VAL("&H"+D$))
1370 SP$=SP$+D$
1380 NEXT I
1390 SPRITE$(J)=SP$
1400 NEXT J
1410 REM Derecha
```

```
1420 DATA 00,00,00,00,1f,1f,06,1f
1430 DATA 1f,1f,06,1f,1f,00,00,00
1440 DATA 00,00,00,00,00,38,10,fc
1450 DATA ec,fc,10,38,00,00,00,00
1460 REM Abajo
1470 DATA 00,00,00,1b,1b,1f,1b,1b
1480 DATA 03,03,0b,0e,0b,03,00,00
1490 DATA 00,00,00,b0,b0,f0,b0,b0
1500 DATA 80,80,a0,e0,a0,80,00,00
1510 REM Izquierda
1520 DATA 00,00,00,00,00,1c,08,3f
1530 DATA 37,3f,08,1c,00.00,00,00
1540 DATA 00,00,00,00,f8,f8,20,f8
1550 DATA f8,f8,20,f8,f8,00,00,00
1560 REM Arriba
1570 DATA 00,00,03,0b,0e,0b,03,03
1580 DATA 1b,1b,1f,1b,1b,00,00,00
1590 DATA 00,00,80,a0,e0,80,80,b0
1600 DATA b0,b0,f0,b0,b0.00,00,00
1610 RETURN
1620 REM Lectura de Datos
1630 RESTORE 1720
1640 FOR J=0 TO 20
1650 READ D$
1660 FOR I=0 TO 22
1670 D=VAL(MID$(D$,I+1,1))
1680 C(I,J)=D
1690 NEXT I
1700 NEXT J
1710 REM  12345678901234567890123
1720 DATA 99999999999999999999999
1730 DATA 9         222         9
1740 DATA 9 99999999111999999999 9
1750 DATA 9 9       222       9 9
1760 DATA 9 9 999999111999999 9 9
1770 DATA 9 9 94    222     9 9 9
1780 DATA 9 9 999999111999 9 9 9
1790 DATA 9 9       222     9 9 9
1800 DATA 9 99999999111999 9 9 9
1810 DATA 93        222  59 9 9 9
1820 DATA 9 99999999111999 9 9 9
1830 DATA 93    222 222    313139
1840 DATA 9 999911199999999 9 9 9
1850 DATA 9 9   222      69 9 9 9
1860 DATA 9 9 9911199999999 9 9 9
1870 DATA 9 9   222         9 9 9
1880 DATA 9 99991119999999999 9 9
1890 DATA 93    222           9 9
1900 DATA 9 9999111999999999999 9
1910 DATA 9     222             9
1920 DATA 99999999999999999999999
1930 RESTORE 2050:CLS
```

```
1940 READ D$
1950 IF D$="fin" THEN 2320
1960 IF D$<>"P"THEN 2000
1970 READ X1,Y1,C1,K1
1980 PAINT(X1,Y1),C1,K1
1990 GOTO 1940
2000 READ X1,Y1,X2,Y2,C
2010 IF D$="L" THEN LINE (X1,Y1)-(X2,Y2),C
2020 IF D$="B" THEN LINE (X1,Y1)-(X2,Y2),C,B
2030 IF D$="F" THEN LINE (X1,Y1)-(X2,Y2),C,BF
2040 GOTO 1940
2050 DATA B,8,7,190,174,3
2060 DATA B,11,10,187,171,3
2070 DATA P,11,8,3,2
2080 DATA B,187,88,254,108,3
2090 DATA B,191,91,251,105,3
2100 DATA P,200,89,3,2
2110 DATA F,186,92,250,104,1
2120 DATA L,26,26,172,26,2
2130 DATA L,172,26,172,156,2
2140 DATA L,172,156,26,156,2
2150 DATA L,26,26,26,74,2
2160 DATA L,26,74,138,74,2
2170 DATA L,138,74,138,90,2
2180 DATA L,138,90,26,90,2
2190 DATA L,138,58,42,58,2
2200 DATA L,42,58,42,42,3
2210 DATA L,42,42,155,42,2
2220 DATA L,155,42,155,138,2
2230 DATA L,155,138,26,138,2
2240 DATA L,42,122,138,122,2
2250 DATA L,138,122,138,106,2
2260 DATA L,138,106,26,106,2
2270 DATA L,26,106,26.138,2
2280 DATA F,42,42,45,58,9
2290 DATA F,135,106,138,122,10
2300 DATA F,135,74,138,90,10
2310 DATA fin
2320 FX=0
2330 FOR J=0 TO 20
2340 FOR I=0 TO 22
2350 XX=I*8+10:YY=J*8+10
2360 IF C(I,J)<>0 THEN 2400
2370 DRAW"BM=XX;,=YY;"
2380 FX=FX+1
2390 DRAW"C8R1"
2400 IF C(I,J)=0 OR C(I,J)>1 THEN 2430
2410 LINE(XX-5,YY)-(XX+6,YY+8),1,BF
2420 LINE(XX,YY-5)-(XX+6,YY+8),1,BF
2430 NEXT I
2440 NEXT J
2450 RETURN
```

El código máquina: ese desconocido

El código máquina o ensamblador representa el lenguaje propio de los ordenadores. Rápido y potente dispone de peculiares características para cada ordenador. Pese a las dificultades que conlleva su aprendizaje, resulta imprescindible su conocimiento para cualquier aficionado a la informática.



Durante mucho tiempo los usuarios de los ordenadores han pensado que el código máquina era un invento extraño que solo lo sabían manejar los tres locos de turno que andaban encerrados en sus "cavernas" informáticas. La extraña sucesión de números en hexadecimal o, en el mejor de los casos, de instrucciones del tipo "LD A, *FF" hacían que la persona que lo veía se asustase pensando qué extrañas órdenes se encerrarían detrás de esos símbolos.

Es de todos sabido que el lenguaje máquina (o ensamblador) es el más rápido, por esta razón los juegos comerciales que invaden el mercado están hechos en este lenguaje. También es un pensamiento popular la creencia de que cuanto más rápido es el "idioma" con el que trabajemos, más difícil

es de aprender. Aunque esto es cierto, no es tan exagerado como muchos creen y una vez aprendido resulta igual de fácil de manejár que muchos otros, sobre todo si se dispone de los útiles adecuados.

Con esta serie que empieza en este número pretendemos dar los conocimientos básicos mediante los cuales aprender a manejar el ordenador a fondo y se puedan manejar sin miedo términos tales como "acumulador", "pila", "banderas", "memoria intermedia" (buffer), etc

Para terminar esta introducción queremos señalar que esta serie será realmente útil si es comprendida por las personas a que va destinada, para lo cual no duden en escribir si nos dejamos algo en el tintero o hay temas que deben explicarse más.

Los ordenadores tienen dieciséis dedos

Esta sorprendente afirmación tiene una razón de ser. ¿Nunca se ha fijado que en las manos tenemos 10 dedos?, y ésta es la base en la que suelen operar los humanos (base 10 o base decimal significa que sólo hay diez valores distintos que se pueden expresar con un dígito: 0,1,2,3,4,5,6,7,8 y 9; el siguiente, el 10, necesita dos dígitos). Pues bién, la mayoría de los ordenadores no operan internamente en base 10, sino en base 16. Esto significa que para ellos existen 16 símbolos distintos para expresar cantidades y, por tanto, las dieciséis primeras cantidades (del 0 al 15 en base decimal para que lo puedan entender nuestros lectores humanos modernos y de 0 a XV para habitantes del Imperio Romano) se pueden expresar con un solo dígito. La siguiente cantidad (16 entre nosotros) necesita dos dígitos.

Para aliviar un poco el dolor de cabeza que, indudablemente, debe tener ya, intentaremos explicarlo de otro modo. Cuando nosotros escribimos sin el menor reparo "159" estamos presuponiendo cosas que, como nuestros lectores venusianos (que operan en base 15) y rigeliano (que operan en base 7) saben, no son tan evidentes. Suponemos que operamos en base 10, lo que significa que el valor de esas cifras es: nueve mas el valor de la segunda (5) multiplicado por la base(en este caso diez) y la tercera (1) multiplicada por la base al cuadrado. De este modo queda 9+50+100=159 ¿no se lo esperaba?. Pero hagamos el mismo proceso suponiendo que la base es dieciséis. Sería 9 más 5 por 16 (la base) más 1 por dieciséis al cuadrado. El resultado es: 9+80+-1024=1113 sorprendentemente, ¿verdad?. Pasemos a un caso mas sencillo: 10 en base decimal es, evidentemente, 10. Pero en base hexadecimal es 0+1*16=16. En todos los casos hechos hasta ahora hemos convertido números en una base a base diez. En algunos casos hemos pasado de base diez a base diez (cálculo bastante inútil como se habrá observado) pero en otros hemos pasado de base hexadecimal (16) a base decimal. A partir de este momento conviene que no se imagine número sino montones de cosas. Si tiene mala memoria imaginativa coja un montón de garbanzos y agrúpelos para hacer los cálculos. Pero tenga cuidado, si quiere calcular la memoria de su ordenador, puede crear una crisis en la industria del garbanzo.

Cuando decimos que un número pertenece a una base, entre sus dígitos sólo pueden estar los comprendidos entre 0 y la base menos 1.

Según el proceso de conversión que hemos explicado, las cantidades existentes entre ningún garbanzo y los que hay entre las llaves siguientes (000000000) (por favor tenga imaginación e imagínese que los círculos son garbanzos) se expresan igual en base decimal y hexadecimal: 0,1,2,3,4,5,6,7,8 y 9. En cambio, una cosa tan sencilla como "10" ya varia. En base decimal significa (0000000000) y en base hexadecimal (0000000000000-000). Supongamos que ya ha comprendido el misterio (si no lo ha hecho, vuelva a leer desde el principio y si aún así no lo consigue, haga un acto de fe y crea en todo lo que decimos a partir de ahora), entonces surge otro problema, ¿como se expresa (0000000-000) en base hexadecimal? Evidentemente no nos vale ninguno de los dígitos que tenemos "10" tampoco es y decir 10-6 no resulta práctico. Para poder expresarlo necesitaremos 6 símbolos mas para expresar esas cantidades que están entre el 9 y el 10 (en base hexadecimal). En lugar de crear una nueva caligrafía, vamos a utilizar algunos ya existentes en nuestra caligrafía: la "A" significa (0000000000) (diez en base decimal), la "B" (000000000-00) a continuación, viene la "C", "D", "E" y, por último, la "F" significa (000000000000000) (15 en base decimal). Sabiendo esto ya podemos operar tranquilamente en hexadecimal. Por ejemplo, si queremos pasar "B3" a base 10, basta aplicar todos los conocimientos aprendidos: 3+B*16 (la base), pero B en decimal es 11, y queda: 3+11*16=179. A partir de ahora y para evitar confusiones de cuando un número se expresa en una base u otra haremos algunas convenciones. En primer lugar, si lo escribimos tal cual significa que está en base diez. En cambio, si

detrás de el ponemos una "h" significa que está en base hexadecimal. De este modo sabremos su significado en un caso u otro. Esto en determinados libros se expresa poniendo el número seguido de un subíndice que indica la base. Así, 13 en decimal se puede expresar como 13_{10} y 13h (o trece hexadecimal) es 13_{16}. Nosotros utilizaremos la notación explicada anteriormente pero recuerde que ambas indican lo mismo.

Antes de seguir se debe hacer una aclaración. Cuando decimos que un número está en una base, entre sus dígitos solo pueden estar los comprendidos desde el 0 al de la base menos uno. Esto quiere decir que si damos un dato en base decimal, solo serán válidos del 0 al 9. A, B, C, D, E y F no tendrán sentido en esta base octal (base ocho) solo serán válidos del 0 al 7. La razón es que si no 10 y A (al no llevar ninguna letra detras se entiende que es en base diez) significarían los dos: (0000000000) por lo que haría una redundancia.

Un sistema rápido para convertir un número en una base cualquiera a decimal es el que se muestra en la figura 1. En ella vemos una serie de cajones numerados a partir del cero empezando por la derecha. Si el dato a convertir lo escribimos en los cajones ajustándolo a la derecha (el dígito de menor peso en el cajón del cero, el siguiente en el del uno, etc.) basta hacer una sencilla operación que se detalla en la figura 2 para el 2A3h. Se coge cada número (o su equivalente en base 10) y se multiplica por la base elevada al número situado encima del cajón, una vez convertidos todos, se suman; el resultado es la expresión en base 10 del dato. De este modo se puede convertir cualquier número por grande que sea.

Pero también se puede pensar que tienen dos dedos

¡¡Horror!!, estará usted pensando, no he acabado de digerir lo de la base hexadecimal y ya me vienen con otra historia. No se alarme, no es tan difícil, en realidad es muy similar a los procesos en base hexadecimal pero con la única diferencia de que operamos en base dos y, por tanto, solo existen en esta base los dígitos 0 y 1 y siendo 1+1=10, ya que no existe el dos y tenemos que cambiar a un número de dos digitos. El sistema de numeración quedaría 0,1,10,11,100,101...

Las conversiones entre bases numéricas es inmediata cuando se trabaja en bases que son potencias de dos.

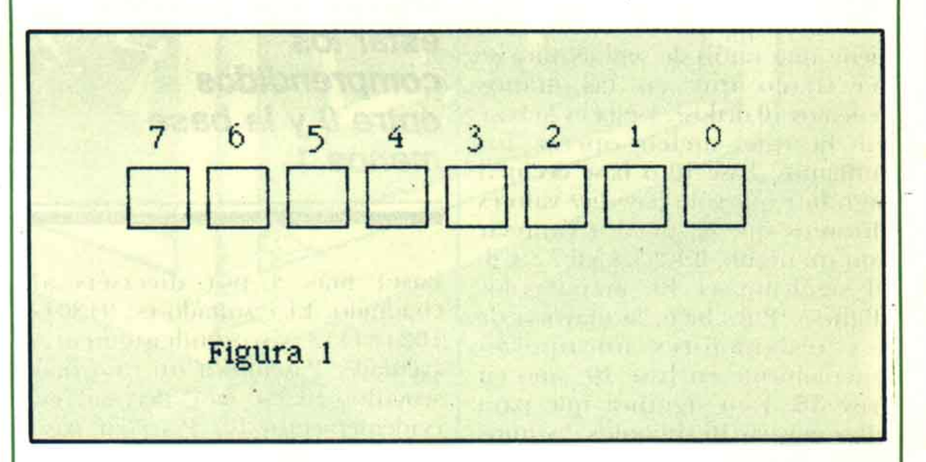


Figura 1

que en decimal son, respectivamente 0,1,2,3,4,5... Para la conversión podemos utilizar el sistema de la caja usado anteriormente, pero teniendo en cuenta que en este caso la **base** es 2. Para diferenciarlo de los otros, consideraremos que un número está expresado en este sistema si después de él hay una "b".

Evidentemente, en él solo tienen sentido las cifras "0" y "1" y no será válida ninguna otra.

Este sistema, denominado binario, es el que realmente usan los ordenadores, pero cuando uno se encuentra con que un número tan simple como el 1018, en binario se expresa como 1111111010, empieza a buscar un método más sencillo de trabajar. Este es el hexadecimal. ¿Y porqué no el decimal? La razón es muy sencilla, si el número anterior lo dividimos en grupos de cuatro números consecutivos empezando por la derecha (0011-1111-1010) y cada grupo le asignamos su equivalente hexadecimal, queda 3FAh, que pasado a decimal queda: $10+15*16+3*(16^2)$ que es 1018, es decir, el número de que partimos. Esta equivalencia entre los números binarios y los hexadecimales se cumple siempre debido a que el 16 es una potencia de 2 (la cuarta) y esto nos permite realizar la agrupación. Pero si intentamos hacer la misma conversión en decimal (que no es potencia de 2), veremos que no se cumple aunque agrupemos los dígitos binarios en cantidades distintas. En realidad la conversión es inmediata para aquellos que sean potencias de dos: 4, 8, 16, 32... y la agrupación en binario se hará con tantos dígitos como indique la potencia respectiva, de este modo, en base 8, que es 2 elevado a 3, tendremos que agrupar de tres en tres, en base 32 sería

de 5 en 5. Debido a normativas del mercado se estableció la base 16 o **hexadecimal como estándar y esa es la que seguiremos, además es la más útil para trabajar con este** ordenador como luego veremos.

Bits, Bytes y otros monstruos secretos

Hay que reconocer que la informática es una de las ramas de la ciendia que más barbarismos usa. Palabras como "debugear" y "linkar" son comunes entre los entendidos. Pero esto se debe a que nuestro idioma no posee términos equivalentes que nos permitan decir lo mismo sin cometer atropellos contra el lenguaje. Muchas veces ni siquiera se españolizan los términos, si no que se usan los ingleses directamente. Bajo riesgo de ser atacados por algún académico de la lengua castellana, vamos a **explicar algunos de ellos que serán usados en esta serie de artículos.**

El primero es *"bit"*, que en ingles significa mordisco, pedazo y que es una abreviatura del término *"Binary digiT"*, o dígito binario. Esta palabra se usa para designar a cada una de las posiciones que ocupan los dígitos binarios. Por tanto, en el número 101101b hay seis *bits*, uno por cada posición. Se usa este término también para representar las capacidades de determinadas memorias. En este caso si decimos, por ejemplo, que una memoria tiene una capacidad de 8 *bits*, estamos expresando que como mucho caben en ella 8 dígitos y, si leemos que se almacena en dicha memoria el número 101101b (de seis cifras) deberemos rellenar a ceros por la izquierda, de modo que quede 00101101b que ya son las ocho indicadas.

Otro término muy importante es el de *"byte"*.Este se usa para indicar una agrupación de 8 *bits* como en el ejemplo que acabamos de ver. De modo que 11010010b es un *byte*, **pero 110b no, dado que solo tiene tres *bits* y 1011011101100111b tampoco, porque tiene 16. La palabra** *byte* nos permite trabajar con grupos más definidos y muy usados en los microordenadores. Por ejemplo, cada una de las memorias de **nuestro ordenador admite un *byte*. así como muchos registros internos, la capacidad de memoria que nos indica el manual, está medida en *byte*.**

Otro término interesante es el de palabra (o *"word"* en ingles) que sirve para indicarnos una agrupación de *bytes*. El tamaño de esta **agrupación depende del ordenador, del programa específico e, incluso,** de la persona que esté hablando. Algunos consideran que una palabra es un *byte*, otros que son dos (16 *bits*) e incluso otros que son 4*bytes* (32 *bits*), a lo largo de esta serie, nosotros consideraremos que son 2 *bytes*, o lo que es equivalente, 16 *bits*.

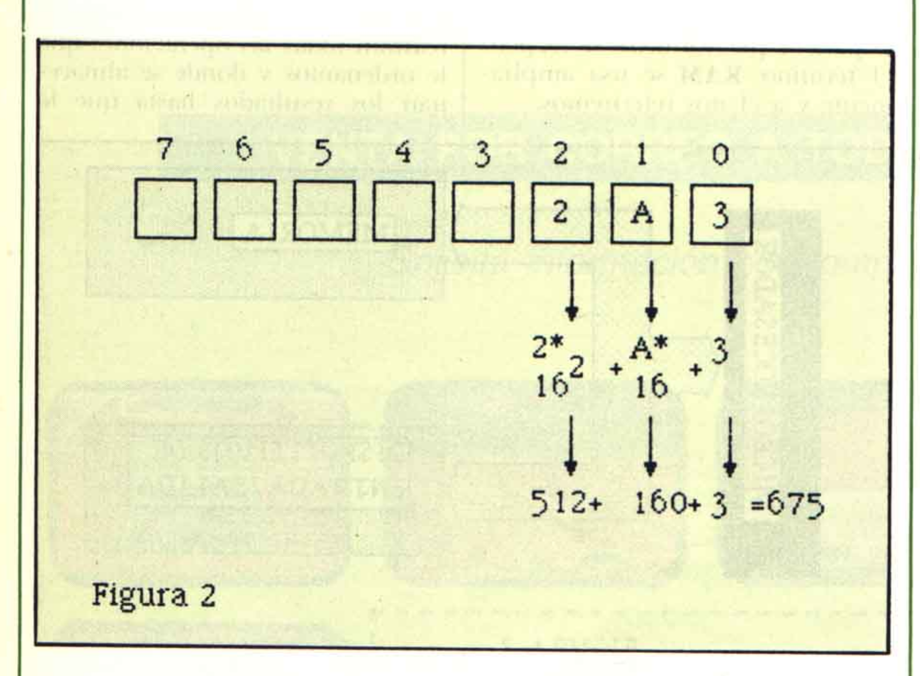


Figura 2

El ordenador por dentro

Debido a que el lenguaje ensamblador está basado en la máquina en que trabaja y se ajusta mucho a ella, es necesario tener algunas nociones de como funciona esta por dentro, pero no se asuste, no vamos a dar aquí un curso de ingenieria rápida.

Nuestro ordenador está compuesto, como puede verse en la figura 3, de tres unidades esenciales. El microprocesador, la memoria y los dispositivos de entrada y salida. Cada uno de ellos tiene una función muy definida que explicaremos con una analogía.

Para ello debe imaginarse que existe una oficina muy moderna que para atender a los clientes tiene a un robot encerrado en un cuarto con una ventamilla y una bandeja que comunican con los clientes. De este modo se libera a la persona correspondiente de esta pesada tarea. Pero el robot tiene un problema, es sumamente estúpido y solo sabe hacer las operaciones básicas, coger un papel sumar dos números, escribir en un sitio o leer de otro. Pero el dueño de la oficina, que es muy inteligente (si están pensando que **esto parece un relato de niños, no lo duden, lo es porque nosotros** somos muy infantiles), ha escrito

en un libro una serie de instrucciones que nuestra emprendedora máquina puede ir leyendo y ejecutando. Estas son del tipo "mira a ver si hay alguién en la ventanilla", "si hay alguién pregunta que quiere", "si no, espera a que venga alguién", etc. De este modo nuestro robot puede hacer una función útil sin necesitar un cerebro mucho más potente como es el de una persona.

El robot de nuestro cuento se llama microprocesador, y existen muchos modelos fabricados por distintas casas que difieren en sus características, unos son más rápidos que otros, algunos saben multiplicar, otros tienen más fuerza y, por tanto, el libro que pueden manejar es más grande, etc. El nuestro (el que lleva su ordenador) se llama Z-80 y comprende unas órdenes muy definidas que es lo que constituye el lenguaje ensamblador. ¡¡¡Un momento!!!, ¿pero mi ordenador no funciona en **BASIC**?, estará diciendo. Efectivamente, pero lo que hace su máquina es tener un libro especial escrito en el lenguaje ensamblador del Z-80 que le dice como traducir las instrucciones de **BASIC** a su lenguaje para poder ejecutarlas. Debido a este proceso de traducción, **el BASIC es más lento y no se pueden hacer muchos juegos con él.**

Los libros y los cuadernos que maneja el Z-80 son lo que denominamos "memoria" y sirven para almacenar la información, ya que él es un olvidadizo. Si son libros escritos a tinta y, en consecuencia, no se puede borrar lo que hay **escrito en ellos, los llamaremos "ROM" abreviatura del término** inglés *"Read Only Memory"* que significa memoria de sólo lectura, que es lo que acabamos de decir. En cambio, los cuadernos están en blanco o como mucho escritos a lápiz y nuestro robot también dispone de lápiz y goma de borrar, por lo que puede hacer correcciones en ellos. A estos los denominamos **"RAM"**, que significa *"Random Access Memory"*, es decir, memoria de acceso aleatorio, que no explica muy bien su uso ya que una definición más propia es memoria de lectura y escritura que es para lo que realmente sirve, pero el término **RAM** se usa ampliamente y a el nos referiremos.

Las tres unidades esenciales de un ordenador son: el microprocesador, la memoria y los dispositivos de Entrada/Salida.

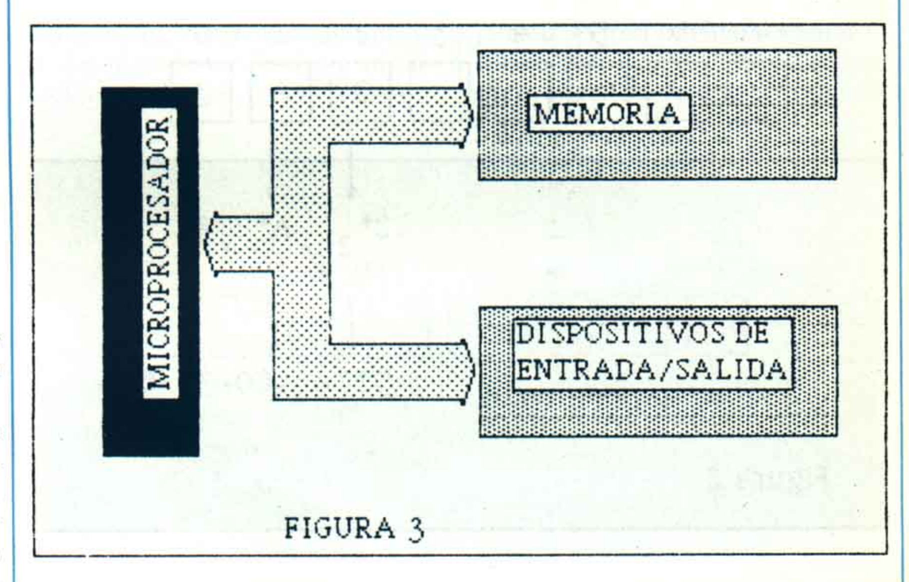


FIGURA 3

Por último nos queda la ventanilla y la bandeja. Estas dos le sirven para comunicarse con el exterior y, por tanto, reciben el nombre de dispositivos de entrada salida, más abreviadamente E S o en inglés, I O.

Todos estos dispositivos los iremos viendo poco a poco, aprendiendo como funcionan y que órdenes tenemos que dar para que actuen correctamente. Como primer paso vamos a ver como está hecho el Z-80.

El corazón de nuestro ordenador: el Z-80

Antes comentamos que el robot era un desmemoriado, y no nos faltaba razón, la memoria que tiene es muy pequeña y está dividida en lo que denominamos registros. Cada uno de éstos permite almacenar un número pequeño y, por si fuera poco, la mayoría tienen unas funciones muy definidas. En la figura 4 podemos ver todos los registros que tiene. El primero de ellos y uno de los más importantes (o por lo menos el más nombrado) es el Acumulador, representado por una "A". Aquí es donde se realizan todas las operaciones que le ordenamos y donde se almacenan los resultados hasta que le mandamos escribirlos en otro lado. En cierto modo, se le puede considerar como una calculadora de mesa que el Z-80 posee para realizar los cálculos que le ordenemos. En ella cabe un *byte* u ocho *bits*, que es lo mismo. Si esto lo pasamos a decimal, veremos que puede almacenar un número entre 0 y 225. (o, como veremos posteriormente, entre -128 y 127) Para operar con números más grandes de-

beremos usar varias operaciones usando un acarreo que veremos luego (¿se acuerda cuando decía "siete más cinco es dos y me llevo una"?, pues el Z-80 lo hace de una forma parecida).

Otro registro importante es el contador de programa. Para referirnos a él usaremos las letras "PC" que son las iniciales de las palabras "*Program Counter*", que significan nada más ni nada menos que "contador de programa". Para comprender su significado debemos volver a la analogía que utilizábamos antes del robot y la oficina. Recordarán que decíamos que tenía todas las órdenes escritas en un libro, pero ¿como se acuerda de porqué página iba?, muy sencillo -dirán- por donde ha dejado el libro abierto. Pero el problema es que al no existir el libro físicamente, no puede dejarlo abierto y necesita tener una marca de la página que está ejecutando. Esta marca es el contador de programa, en el que se almacena el número de la página. Dependiendo de el tamaño de este contador, podrá manejar mas o menos memoria, ya que sería totalmente inútil tener una memoria a la que no podía referirse ni acordarse de ella. En definitiva no podría usarla. En nuestro caso, el contador de programa admite números de dieciseis bits, o dos bytes, o una palabra, por lo que podrá manejar todas las memorias (páginas) entre la cero y la 65535, en hexadecimal entre 0h y FFFFh, o si lo prefiere en binario entre 0000000000000000h y 1111111111111111h. Algunos de los que estén leyendo pensarán que nos hemos equivocado, ya que hay ordenadores que usan el Z-80 y manejan más memoria, esto es cierto pero para ello deben recurrir a técnicas complicadas como por ejemplo el paginado que consiste en sacar un bloque de páginas del libro y sustituirlo por otra, de modo, que el máximo nunca sobrepase el 65536. Debemos señalar que en cada una de las posiciones apuntadas (memoria de un byte) solo cabe una instrucción y después de leerla el Z-80, el contador de programa se incrementa automáticamente en uno de modo que cuando tenga que ejecutar la siguiente sólo necesite mostrar esa dirección.

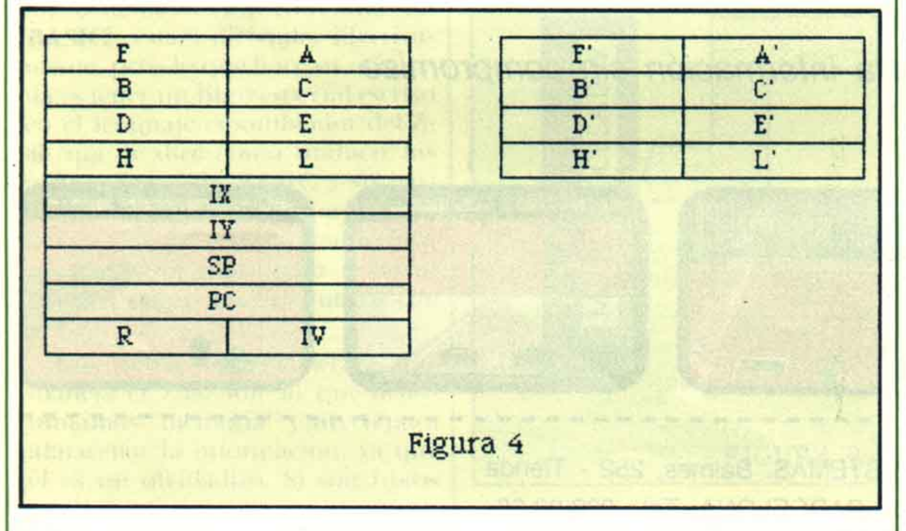


Figura 4

Además de los registros antes mencionados, existen otros que utiliza como memorias temporales para almacenar resultados intermedios (pero que no admiten todas las operaciones que se pueden realizar con el acumulador). Estos son los siguientes: B, C, D, E, H, L. Todos ellos son de un byte de longitud pero poseen la curiosa propiedad de poder funcionar en determinadas ocasiones como parejas, de modo que forman un solo registro de 16 **bits** (2 *bytes*). las parejas las forman del modo siguiente: B y C, D y E, y H y L. Más adelante veremos los usos que se les pueden dar a todos ellos.

Existe otro tipo de registros especiales que son los denominados de índice. Existen dos de ellos, denominados respectivamente IX e IY y ambos son de dieciseis bits. Para comprender su utilidad basta recordar algo que nos ha pasado muchas veces para referirnos, por ejemplo, a un coche, a veces decimos "es ese", pero otras, sobre todo cuando está más lejos, es "el tercero" y en algunos casos la expresión es "el tercero a partir del rojo". En este último caso utilizamos otro coche de referencia a partir del cual indicamos la posición. En el Z-80, la posición del coche rojo puede estar almacenada en el registro del índice, lo que simplifica muchos cálculos como veremos más adelante.

Para poder aprender el uso del registro SP ("*stack pointer*" o puntero de pila) que es otro registro de 16 bits, debemos comprender primero lo que es una pila (y no nos estamos refiriendo a esos cilindros que se le ponen a las radios y linternas). Estas son muy comunes en toda nuestra vida aunque no las designemos por ese nombre. Un caso típico es cuando amontonamos papeles y luego los vamos cogiendo en orden inverso al que usamos al amontonarlos. El microprocesador nos permite realizar esta operación con los números de modo que si queremos guardar momentáneamente un grupo elevado de números,podemos evitarnos la necesidad de buscar una memoria libre metiéndolos en la pila. Pero como comentamos antes, el ordenador solo posee páginas numeradas, de modo que lo que hace es tener un puntero, similar en algunos aspectos al contador de programa, de modo que

cuando le damos la orden de guardar un número en la pila, lo guarda en la posición indicada por ese puntero,que incrementa después para que el siguiente número lo almacene en la memoria siguiente. Al recuperar el número el proceso se hace a la inversa, restándole uno a este puntero y recuperando el contenido de la memoria apuntada después.

El registro que en el gráfico 4 aparece al lado del acumulador es el "F" ("*Flags*" o banderas) que es de ocho bits, pero a diferencia de los demás no está pensado para almacenar números (aunque teóricamente podría) y cada uno de sus bits no indica un valor sino una condición.

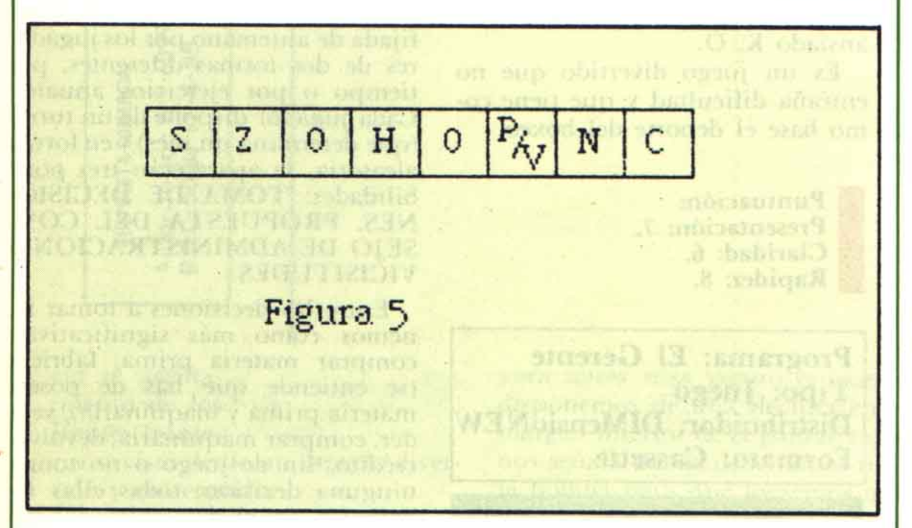


Figura 5

En la figura 5 vemos que cada uno de ellos tiene asignada una inicial que corresponde a un nombre. Cuando el ordenador mira este registro, no lo hace en conjunto si no que mira "si este bit está a uno" o "si el otro está a cero". Una descripción muy sucinta de ellos (ya que más adelante les dedicaremos más atención) sería: el *bit* O (el situado más a la derecha) es el "C" o de acarreo. Este es el que guarda el uno de más cuando, por ejemplo, hacemos una suma (¿recuerda lo de 7 mas 5 es 2 y me llevo 1?, pues el uno se almacena aquí). El siguiente *bit* (el 1) es el "N", y es usada por el microprocesador para saber cuando ha hecho una suma o una resta. Es usada internamente y no tiene ninguna utilidad a la hora de programar. El tercer *bit* (el 2 según el sistema de numeración que usamos) es el "P V" y tiene dos funciones distintas. Cuando se están realizando operaciones aritméticas con signo indica si se ha producido un "*overflow*" o lo que es lo mismo, un rebosamiento de la capacidad del acumulador. En este sentido es algo similar a la bandera de acarreo pero aquella funciona con números sin signo y esta lo hace con números con signo (una disertación más amplia sobre estos dos tipos de operaciones se hará más adelante, de momento solo acuérdese de que existen estas dos banderas). Si lo que se ha realizado es una función lógica (que también veremos más adelante) se nos mostrará aquí la paridad del resultado. Si hay un número par de unos (0,2,4,6 y 8), esta bandera estará a uno, de lo contrario estará a cero. El *bit* 3 no se usa y siempre está a cero El *bit* 1 es el "H" o de medio acarreo e indica lo mismo que el de acarreo pero cuando la suma de los *cuatro* primeros bits (no los ocho) da un quinto de exceso. El *bits* 5 tampoco se usa y de un modo similar al 3 siempre está a cero. El *bit* 6 es el de cero ("Z") y se pone a uno cuando de resultas de una operación en el acumulador no hay ningún uno, sino que todos los *bits* están a cero. Por último, el *bit* 7 es el de signo ("S") y se usa solo con las operaciones aritméticas con signo. Su valor es siempre igual al del *bit* más significativo del acumulador (el 7 ya que empezamos a numerar desde el cero).

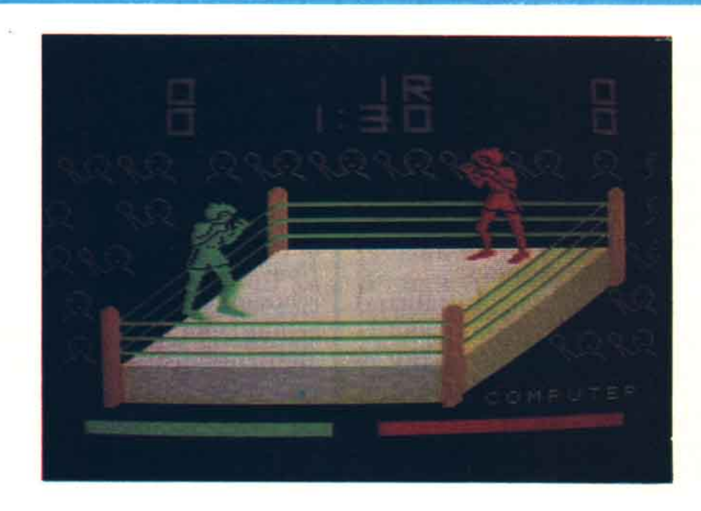
Existen otros dos registros con funciones muy especiales que de momento no son de interés por lo que los comentaremos muy por encima. El primero de ellos es el denominado registro R. Este solo es utilizado por el Z-80 para generar la señal de refresco de las memorias dinámicas (¿le ha sonado a chino?, no se preocupe, no necesita saber más que es eso a menos que quiera ampliar su cultura general). El otro registro es el IV, y es usado durante las interrupciones (determinadas señales que le dicen al Z-80 que deje lo que está haciendo y se ponga a ejecutar otro programa) y no le manejaremos hasta dentro de bastantes capítulos.

También habrá notado que hay unos registros denominados A', F', B', C', D', E', H' y L', estos son similares a sus homónimos sin comillas pero no se pueden usar a menos que demos una instrucción especial, en cuyo caso se cambiarán unos por otros. Se suelen usar en ocasiones especiales en las que queramos guardar el contenido de aquellos registros muy rapidamente para realizar algunas operaciones.

En el próximo capítulo

Una vez echa una descripción del modo de funcionamiento del ordenador y de la estructura interna del microprocesador, en el próximo capítulo empezaremos a ver las primeras instrucciones que comprende y haremos algún programa que se pueda meter en la máquina. □

Programa: Heavy Boxing.
Tipo: Juego.
Distribuidor: E.M.S.A.
Formato: Cartucho ROM.



Tenemos ante nosotros, una cuidada videotización de lo que es un combate de boxeo.

Es una de las mejores maneras de hacer deporte en casa teniendo en cuenta que nuestro adversario en el juego cuenta con capacidad combativa de un campeón de los pesos pesados. El cuadrilátero tan conocido para nosotros en los grandes combates, podemos conseguir tener una expectativa casera, entablando una dura pugna, entre la máquina y nuestras modesta capacidad luchadora, referido a los dos jugadores de la ardua trama combativa.

La composición del juego, es fácil desde nuestro joystick que nos hace vivir intensamente, nuestros movimientos, reflejos y rapidez que conllevan tan importantemente cualquier deporte y como no, especialmente el boxeo, el cual necesita la misma compenetración cuerpo- mente como en este caso joystick-jugador.

La instrumentación del juego es de una rápida comprensión, el joystick nos permite mover al jugador y llevar a cabo golpes que nos permitan derrotar al contrario, pulsaremos el botón de disparo, lanzando así un golpe que puede ser o no acertado dependiendo de la posición del contrario.

La infraestructura del juego es como la de cualquier combate de boxeo, se divide en doce asaltos con descenso entre ellos, a su vez un tiempo de diez segundos que se le da a cada jugador en caso de caer al suelo y si ocurre como es de esperar en todos los combates de boxeo al ansiado K. O.

Es un juego divertido que no entraña dificultad y que tiene como base el deporte del boxeo.

Puntuación:
Presentación: 7.
Claridad: 6.
Rapidez: 8.

Programa: El Gerente
Tipo: Juego
Distribuidor: DIMensioNEW
Formato: Cassette

¿Quién no se ha sentido tentado por el deseo de ser un gran director de empresa?, ¿quién no ha deseado saber en alguna ocasión lo que se siente cuando uno toma la decisión económica de un cierre patronal, de pretender lanzar un producto nuevo o de afrontar una huelga general? Pues bien, con este juego de simulación empresarial vas a tener la posibilidad de medir tu capacidad individual en la gestión de tipo industrial. Pensado para participar hasta seis jugadores, nuestra tarea consiste en la dirección de una empresa que posee una fábrica vacía y 1000$ para invertir. El objetivo es claro: máximo beneficio al finalizar el ejercicio anual.

La duración del juego puede ser fijada de antemano por los jugadores de dos formas diferentes, por tiempo o por ejercicios anuales. Cada jugador dispone de un turno (que determina un mes) y en forma aleatoria, le aparecerán tres posibilidades: TOMA DE DECISIONES, PROPUESTA DEL CONSEJO DE ADMINISTRACION o VICISITUDES.

Entre las decisiones a tomar tenemos como más significativas, comprar materia prima, fabricar (se entiende que has de poseer materia prima y maquinaria), vender, comprar maquinaria, devolver crédito, fin de juego o no tomar ninguna decisión, todas ellas repercuten en el desarrollo del balance de tu empresa. Como consejo, es importante poseer un remanente de dinero suficiente para hacer frente a las posibles vicisitudes que imprevisiblemente pueden aparecer en tu marcha como ejecutivo y que pueden afectar el desarrollo de tu empresa de tal manera, que de no ser previsor es posible dar una quiebra.

La variedad de posibilidades, la contínua información que uno recibe, indispensable en el buen desarrollo de la gestión junto a la multitud de vicisitudes de que puede ser objeto, hacen de este juego un programa atractivo para los amantes de la gestión industrial.

Si existe algún punto criticable

este será la práctica inexistencia de gráficos, haciendo el juego algo monótono en su evolución.

Puntuación:
Presentación: 6
Claridad: 7
Rapidez: 7

Programa: Cannon fighter.
Tipo: Juego.
Distribuidor: E.M.S.A.
Formato: Cassette.

Nos encontramos frente a un juego singular por su dinamismo, entretenimiento e incluso por su carácter innovador. Debemos tener en cuenta que el tema o base del juego es la guerra pero la guerra no vista en el sentido tan amplio de la palabra, sino en un aspecto más conciso y directo como es la defensa de un fuerte.

La defensa de un territorio el asentamiento de los límites fronterizos, visto desde un aspecto divertido y entretenido.

El croquis de juego se establece en tres frentes consecutivos, en el espacio seco y árido del desierto el cual podemos advertir por los colores que nos muestra el programa.

Los tres frentes son defendidos por el "cannon fighter" o cañón solitario que tras las murallas establecidas a lo largo de la pantalla defienden la llegada al fuerte con la diligente disposición de que seamos capaces de desarrollar en tan ardua lucha.

La batalla se desarrolla frente a 5 tanques, que avanzan en formación y que sólo pueden ser destruidos por nuestra pericia y habilidad al lanzar las bombas sobre ellos, para hacer más seguro el acierto disponemos de una flechita en el margen inferior de la pantalla que nos señala donde deberíamos tirar la bomba para que logremos una baja en el enemigo pero a su vez



hay que tener en cuenta el movimiento uniforme acelerado de los tanques y la capacidad armamentística de estos contra nuestras barreras, e incluso si la mella en estas es muy importante contra nosotros mismos.

A su vez la destrucción de nuestra meta no se consagra al haber aniquilado a los 5 tanques, pues más atrasados aparecen nuevas unidades que deben ser también objetivo de nuestros tiros. El constante fuego al que nos vemos expuestos nos hace movernos a una velocidad importante a lo ancho de la pantalla bordeando interiormente nuestra barrera. Una vez agotadas las sucesivas barreras ya que el fuego enemigo ha conseguido su parcial o total destrucción, llegamos a lo, que es el objetivo de nuestra defensa, el cual en caso de derrota se verá coronado por la bandera del enemigo.

Para cubrir nuestra meta, es decir, impedir la toma del fuerte debemos desarrollar un fuego intenso el cual puede variar desde cañonazos a larga distancia o una senda ametralladora para cuando los tanques enemigos se acerquen a las barreras.

Es un juego que cuida el detalle como podemos ver al estar reflejadas en el fondo de la pantalla las Pirámides de Egipto y también el vuelo que desarrollan en el margen superior de la pantalla un ciclo ininterrumpido de aviones. La movilidad de nuestro muñequito, brazo derecho de nuestra defensa nos hace grata la visualización del programa.

Es un juego detallado y con una gran posibilidad de entretenimiento, si queremos pasar un rato bélico sin consecuencias.

Puntuación:
Presentación: 8.
Claridad: 7.
Rapidez: 8.

Programa: Mr. Ching
Tipo: Juego
Distribuidor: E.M.S.A.
Formato: Cartucho ROM

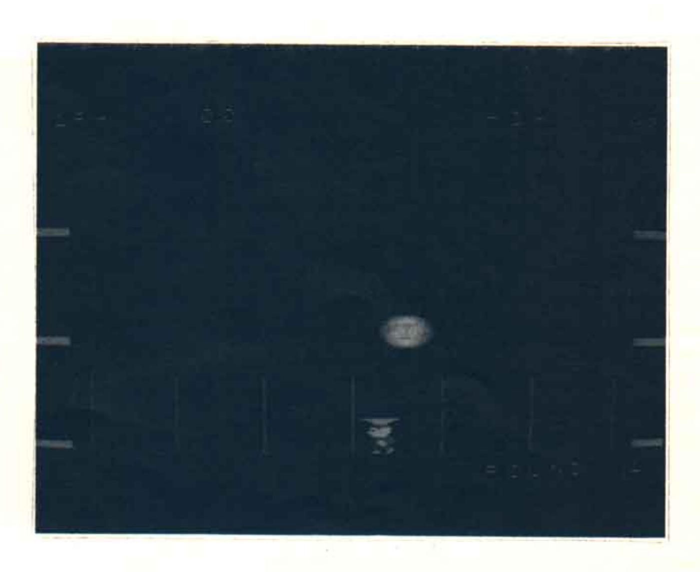
Dentro de la extensa gama de software podemos encontrar para MSX nos encontramos con un entretenido juego que los videoadictos les puede causar furor.

La capacidad dinámica reflejada en la velocidad y audacia nuestra nos hará pasar un rato entretenido y ameno.

Las características del juego versa en la acción malavarísta del chino Mr. Ching que en este caso controlamos desde nuestro joystick, y su tarea un tanto difícil de colocar un número definido de platos sobre palos haciéndolos girar sin que estos pierdan su ritmo.

No obstante la dificultad que entraña el juego, se puede subsanar con una pequeña diversificación de tareas. La adquisición de platos la hacemos a los laterales de la pantalla, en la que existe una pequeña tarima, donde debemos subirnos para recogerlos. Rápidamente al darnos el plato debemos de correr a dejarlo encima de uno de los palos, hasta que completemos los siete del primer piso, una vez conseguido esto y sin dejar que los platos puestos se paren, uno de ellos, iniciará un ascenso al piso superior para así conseguir completar todos los palos de los diferentes pisos, pero ¡un momento!, no pienses que este es el único atractivo del juego, su genialidad, se encuentra en las adversidades que nos presenta y las cuales debemos de ser capaces de resolver, las adversidades antes mencionadas se nos presentan, en este juego, con la aparición de una figura diabólica que nos tira toneles y cuchillos, que habremos de evitar, saltando, agachándonos o bien subiendo a uno de los palos mientras que depositamos un plato que además es la manera más eficiente de no perder el tiempo y probar nuestras habilidades.

Puntuación:
Presentación: 8.
Claridad: 8.
Rapidez: 8.

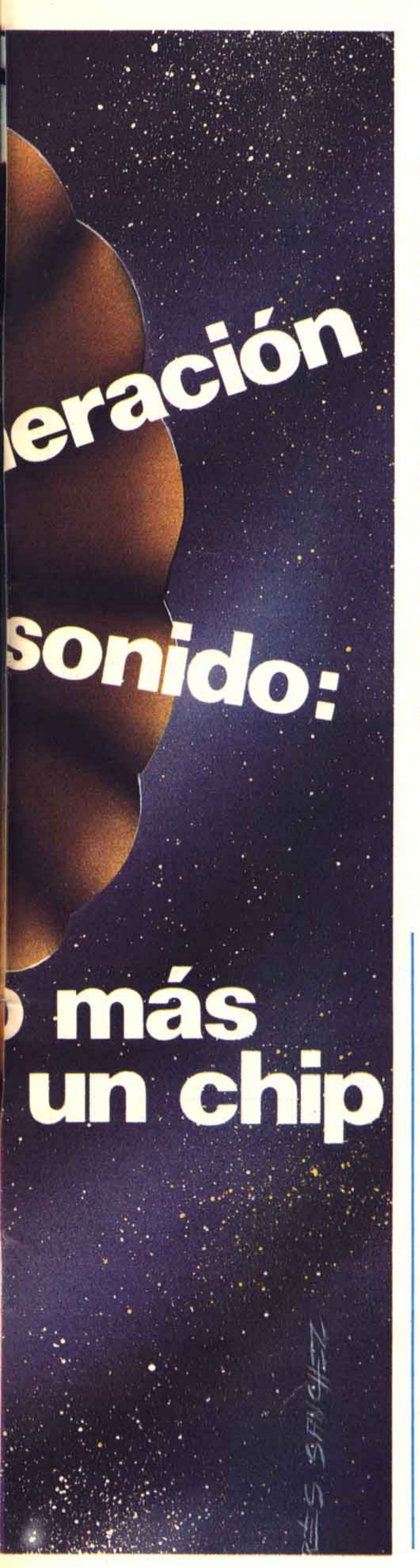



GANE
7.000 PTAS.
todos los meses
PARTICIPANDO
EN NUESTRO
CONCURSO
A partir del próximo número MSX premiará mensualmente los programas que hagan llegar los lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.

Ge

de

alg
que



Recuerdo aquellos viejos tiempos, cuando yo me iniciaba en el mundo del ordenador, en los que el sonido era un sueño lejano, del que tan sólo sabíamos que sería una realidad, pero que era tan complejo que ni siquiera nos atrevíamos a pensar en él.

En aquel entonces, los aparatos que usábamos sólo hacían algún PIIP o TING espúreo acompañando a algún error grave o a la puesta en marcha. Los que más nos arriesgábamos, conseguíamos incluir algún TUIT en nuestros programas, e incluso hubo locos que consiguiero hacer sonidos que nos parecían (esforzando la imagición) disparos de extraterrestres contra nuestra nave.

Y entonces llegó el "chip" AY—3—8910.

De la ronquera a la polifonía

Este circuito integrado tiene la particularidad de disponer de tres canales de sonido (musicalmente hablando, tres voces) con ajustes de tono, volumen y envolvente de volumen independientes, además de un canal de ruido direccionable a 0, 1, 2 o los tres canales de sonido. ¿Y qué quiere decir toda esta jerga técnica?

Cuando uno toma en sus manos una flauta y hace una escala puede tocar una sóla nota cada vez. Hagamos esto con nuestro MSX: PLAY "cdefgab" «RETURN».

Si hemos ajustado bien el volumen de nuestro receptor de TV, oiremos el clásico Do-Re-Mi-Fa-Sol-La-Si. Esto lo hemos realizado mediante el comando PLAY (que quiere decir "toca", "suena") admite a continación una serie de notas y otros comandos que hacen sonar al AY—3—8910. Las notas que hemos escrito están en notación americana, en la que:

A = LA B = SI C = DO
D = RE E = MI F = FA
G = SOL

Pero supongamos ahora que lo que tocamos es un piano. Podemos hacer sonar simultáneamente hasta 10 notas (o más, si hacemos un poco el bruto). Pues bien, los **MSX** son como un piano tocado con 3 dedos. Cada uno de estos dedos se denomina técnicamente "VOZ" o "CANAL". Hagamos una prueba:

PLAY "r8cdefgab", "r8bagfedc", "o5abababa" «RETURN»

Suena mucho más bello y sinfónico, ¿verdad?. Las comas que vemos entre los distintos grupos sirven para decirle al ordenador que el primer grupo lo va a tocar el primer canal (canal A), que el siguiente grupo va a sonar en el canal E, y que el último sonará en

el canal C (tercer canal). De este modo conseguimos que los tres grupos de notas suenen a la vez. En este caso hemos utilizado, además de las notas (a-g), unos comandos, "r" y "o". El comando "o" cambia la octava en la que trabajamos. Expliquemos esto. Supongamos que no tenemos más notas que las que van de Do a Sí. Son tan sólo 8 notas, y cubren muy pocos sonidos (exactamente 8). Si quisiéramos generar sonidos más agudos o más graves, no podríamos. Para ello se ha creado este comando, "o", que cambia la octava de los sonidos siguientes, donde una octava es el intervalo sonoro que hay entre Do y Sí. Normalmente, si no le decimos nada al ordenador, trabajamos en la octava 4. Lo que hemos hecho al escribir "o5" es decirle que toque en la octava 5 (más aguda).

El otro comando empleado, el comando "r", merece explicación aparte. Supongamos que estamos componiendo la Banda Sonora Original de una película de suspense, entonces se, nos ocurre que, cuando el asesino está a punto de matar a la chicha, sería interesante que la música callase por un breve período. Para ello existe este comando, que deja al generador de sonidos en suspenso durante un breve período, marcado por el número que le sigue. Volveremos sobre ello cuando hablemos de las longitudes de las notas. En cualquier caso, habéis de saber que el pequeño silencio insertado al comienzo de las secuencias de sonido de los dos primeros canales sirve para que suenen los tres canales a la vez, ya que, si no, se descompensaría la armonía al perder el canal C un cierto tiempo en interpretar el comando "o5" que tiene el principio.

El chip AY-3-8910 es el "grupo musical" de los ordenadores MSX.

Cómo modificar las notas

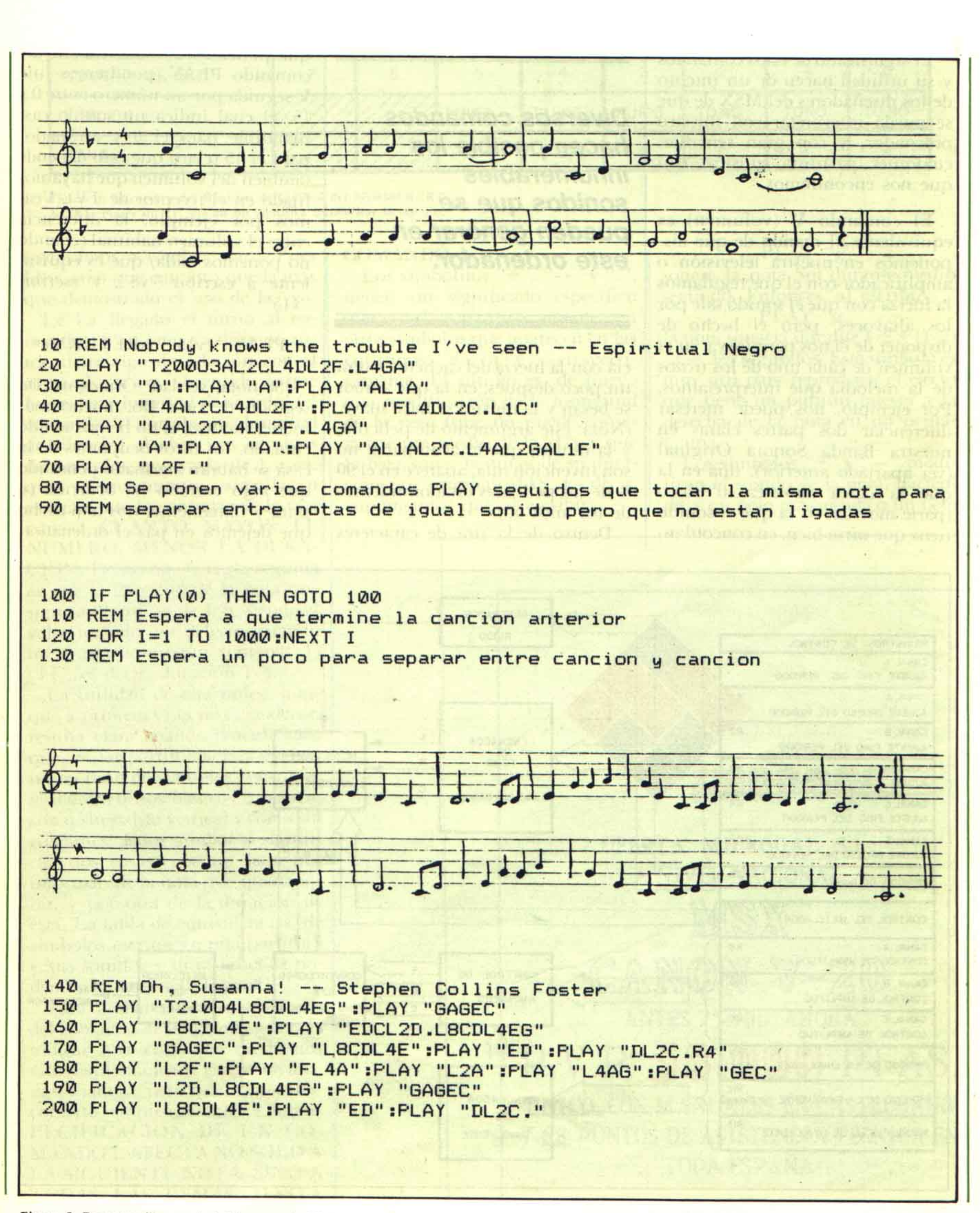
Vamos a seguir ahora analizando las posibilidades del comando PLAY a través de los subcomandos, tales como "o" y "r", de que dispone para modificar el significado puro y simple de las notas que escribamos. Para ello vamos a estudiar, no todos, sino los comandos más importantes, que son:

V = Volumen
L = Longitud + = sostenido — = bemol = puntillo (el mismo significado que en música)
T = tiempo

NOTA	COMANDO "L"	NOMBRE	SILENCIO CORRESPONDIENTE
𝅝	L 1	REDONDA	𝄻
𝅗𝅥	L 2	BLANCA	𝄼
♩	L 4	NEGRA	𝄽
♪	L 8	CORCHEA	𝄾
𝅘𝅥𝅯	L 16	SEMICORCHEA	𝄿
𝅘𝅥𝅰	L 32	FUSA	𝅀
𝅘𝅥𝅱	L 64	SEMIFUSA	𝅁
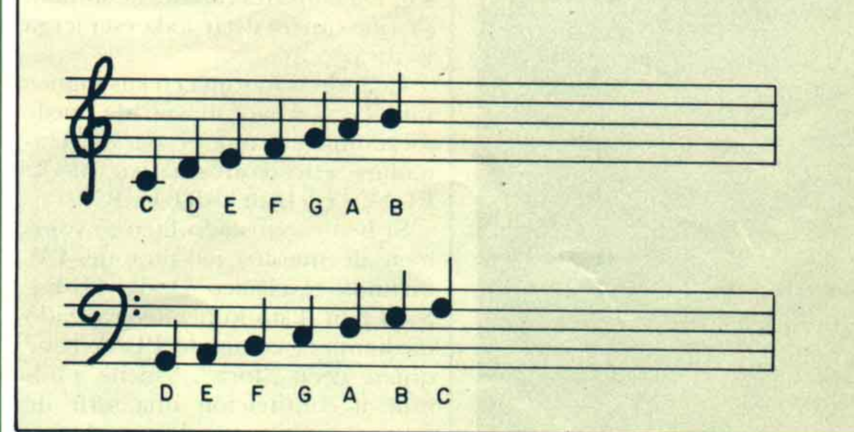


Figura 1: Equivalencia entre notas musicales y longitudes.



```
10 REM Nobody knows the trouble I've seen -- Espiritual Negro
20 PLAY "T200O3AL2CL4DL2F.L4GA"
30 PLAY "A":PLAY "A":PLAY "AL1A"
40 PLAY "L4AL2CL4DL2F":PLAY "FL4DL2C.L1C"
50 PLAY "L4AL2CL4DL2F.L4GA"
60 PLAY "A":PLAY "A":PLAY "AL1AL2C.L4AL2GAL1F"
70 PLAY "L2F."
80 REM Se ponen varios comandos PLAY seguidos que tocan la misma nota para
90 REM separar entre notas de igual sonido pero que no estan ligadas

100 IF PLAY(0) THEN GOTO 100
110 REM Espera a que termine la cancion anterior
120 FOR I=1 TO 1000:NEXT I
130 REM Espera un poco para separar entre cancion y cancion

140 REM Oh, Susanna! -- Stephen Collins Foster
150 PLAY "T210O4L8CDL4EG":PLAY "GAGEC"
160 PLAY "L8CDL4E":PLAY "EDCL2D.L8CDL4EG"
170 PLAY "GAGEC":PLAY "L8CDL4E":PLAY "ED":PLAY "DL2C.R4"
180 PLAY "L2F":PLAY "FL4A":PLAY "L2A":PLAY "L4AG":PLAY "GEC"
190 PLAY "L2D.L8CDL4EG":PLAY "GAGEC"
200 PLAY "L8CDL4E":PLAY "ED":PLAY "DL2C."
```

Figura 2: Demostración musical del comando PLAY.

El significado de estos comandos y su utilidad nacen de un intento de los diseñadores del MSX de que se pueda interpretar con nuestro ordenador, y con gran sencillez, cualquier partitura musical con que nos encontremos.

El comando V (volumen) es equivalente al mando de que disponemos en nuestra televisión o amplificador con el que regulamos la fuerza con que el sonido sale por los altavoces, pero el hecho de disponer de él nos permite variar el volumen de cada uno de los trozos de la melodía que interpretamos. Por ejemplo, nos puede nteresar diferenciar dos partes claras en nuestra Banda Sonora Original (ver apartado anterior), una en la que la chica abofetea al chico (parte movida, en la que el sonido tiene que oirse bien, en concordancia con la fuerza del cachete), y otra un poco después, en la que ambos se besan y la música se hace suave. (Nota: este argumento de película, y la música que lo acompaña, no son invención mía, aparece en el 90 % de las películas estadounidenses de los años 50).

Diversos comandos hacen posible los innumerables sonidos que se pueden generar en este ordenador.

Dentro de la tira de caracteres que ponemos a continuación del comando PLAY, pondremos una V seguida por un número entre 0 y 15, el cual indica un sonido casi inaudible para el 0 y atronador para el 15 (claro que esto depende también del volumen que hayamos fijado en el receptor de TV). Veamos por ejemplo, la diferencia entre el volumen habitual (cuando no ponemos nada) que es equivalente a escribir "v8", y escribir "v15":

PLAY "cv15c" «RETURN»

¿Se nota, verdad? Oímos un Do seguido de otro Do mucho más fuerte, incluso, si no hemos tenido cuidado, los otros ocupantes de la casa se habrán alarmado pensando que algo extraño ha ocurrido (o pura y llanamente, nos han dicho que dejemos en paz el ordenador,

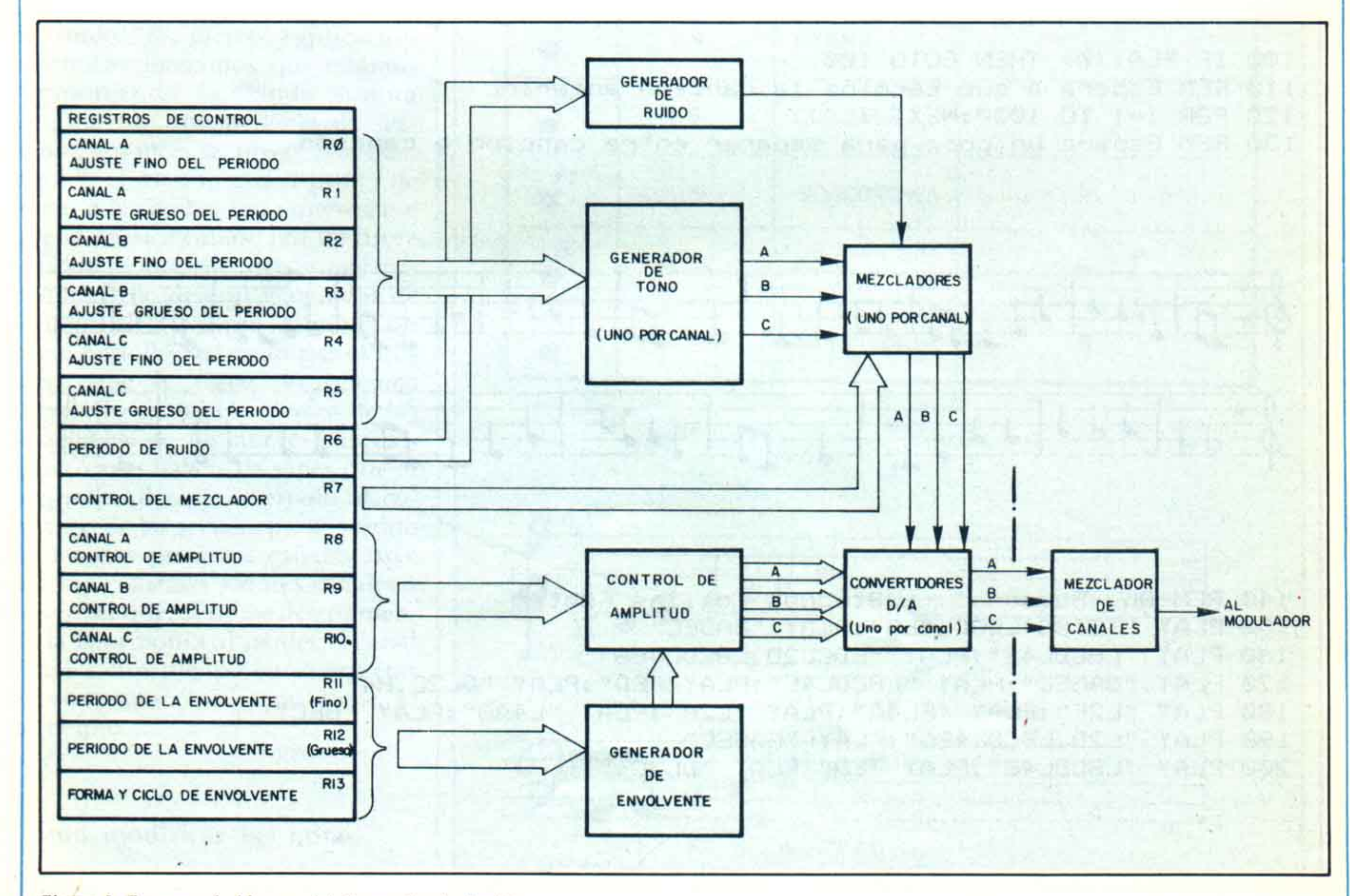


Figura 3: Esquema de bloques del Generador de Sonidos.

NUMERO DE BIT	7	6	5	4	3	2	1	0
FUNCION			RUIDO	RUIDO	RUIDO	TONO	TONO	TONO
CANAL			C	B	A	C	B	A

Figura 4: Estructura del Registro de Control del Mezclador (R7).
NOTA: Poner un "1" en un bit significa inhabilitar la opción.

que vamos a romper algo). Pues bien, creo que con ésto queda más que demostrado el uso de la "v".

Le ha llegado el turno al comando "l". Como ya dijimos antes, tiene algo de relación con el comando "r", en el sentido de que fuerza una longitud de sonido (el comando "r" forzaba una longitud de silencio). Ambos van seguidos de un valor que varía de 1 a 64, que, curiosamente, significan exactamente lo contrario de lo que parece: CUANTO MAYOR EL NUMERO, MENOR LA DURACION. De hecho, la regla seguida es que la longitud de la nota que sigue a "l«n»" es de 1/n, siendo el valor tomado por defecto (cuando no se escribe ningún comando l) "14", es decir, duración 1/4.

La utilidad de esta orden, aunque a primera vista no es evidente, resulta clara cuando descubrimos que, en las partituras, hay escritos un montón de símbolos consistentes en circulitos blancos o negros, con o sin palito vertical y con o sin colgantes. Estos símbolos significan dos cosas: por un lado nos informan de la nota que queremos oír, y por otra de la duración de ésta. La tabla de equivalencias de símbolos escritos en una partitura y sus sonidos y duraciones la podemos hallar en la figura 1.

Para acabar con el comentario del comando de longitud nada más indicar una cosa que no hemos visto explícitamente escrita en ningún manual, pero que hemos descubierto empíricamente. LA ESPECIFICACION DE UN COMANDO L AFECTA NO SOLO A LA SIGUIENTE NOTA, SINO A TODAS LAS DEMAS, HASTA QUE APAREZCA UN NUEVO COMANDO L.

Los simbolitos "+", "—" y ".", tienen un significado específico que se relaciona directamente con otros símbolos que aparecen en las partituras: "#", "b" y ".", respectivamente. El último es, de nuevo una modificación de la loongitud de la nota (que no puede conseguirse por medio de la "l"), y que la hace sonar la mitad de su duración por original MAS. Es decir, supongamos que al escribir:

PLAY "g"

sonase la nota Sol durante medio segundo. Entonces, al escribir

PLAY "g."

la nota sonaría durante 1/2 + (1/2) /2 = 3/4 segundos. Este símbolo es muy útil para reproducir el efecto que tiene un puntito puesto a la derecha de la nota en un pentagrama.

Los símbolos "+" y "—" modifican el sonido de la nota, haciéndola un pelo más aguda o un pelo

más grave, respectivmente, de lo normal. Por ejemplo, si escribimos "d—" o "c +" (que suenan igual) queremos decirle al ordenador que haga un sonido que esté a mitad de camino entre "c" y "d". Este sonido se denomina en música "Re bemol" o "Do sostenido". Hay, sin embargo, dos notas que no es necesario añadirles estos simbolitos. Estas notas son: "b" que, al añadirle el " + " suena exactamente igual que "c" y, como es lógico, "c" que, al ponerle un "—" suena como la "b". Esto no es una peculiaridad propia del **MSX**, sino de toda la música desde que, un poco antes del Renacimiento, se crearon las normas que, aún hoy sigue la música.

Y ahora una pregunta para todos aquellos que alguna vez hayan intentado comprender el significado de una partitura: ¿os habéis fijado alguna vez en unas palabritas que aparecen escritas al principio de las partituras, justo encima del pentagrama? Estas palabras suelen ser "Allegreto", "Andante", "Moderato"... y no tienen nada que ver con el estado de ánimo del autor al componer la canción, ni con el ejercicio físico que prefería. Son unas indicaciones que se le dan al intérprete sobre la velocidad con que ha de ejecutar la pieza. Estas indicaciones son suficientemente flexibles como para que cada cual toque la canción como mejor le suene, pero, de todos modos, resulta conveniente saber que "Prestissimo" significa muy rápido, "Moderato" muy lento, y en medio quedan categorías como "Presto", "Allegretto", "Allegro", "Andante", "Andantino"... además de poder sufrir el efecto de adjetivos como "ma non troppo" ("pero no demasiado"), etc.

Para interpretar esto desde el punto de vista del MSX, disponemos de otro comando, "t" que corresponde a la palabra italiana "tempo", que es el nombre que se le da a esa indicación que hemos mentado en el párrafo anterior.

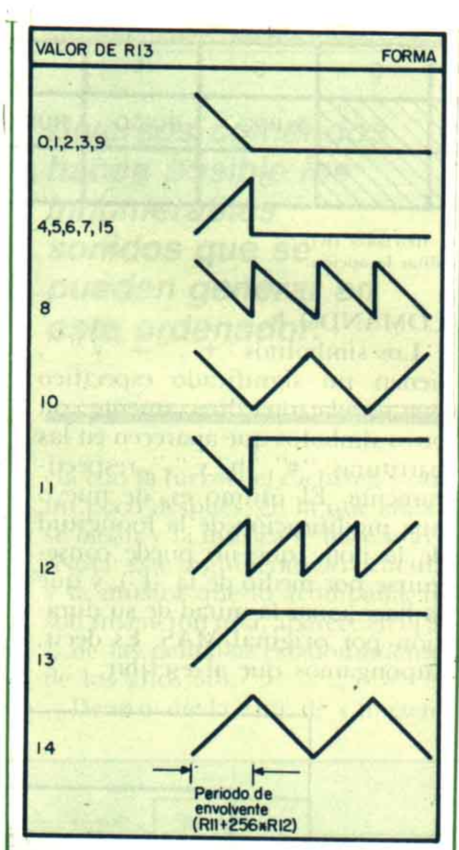


Figura 5: Formas de envolvente.

Pero el comando del MSX no se utiliza diciendo "tallegro", ya que esto es muy subjetivo, y un ordenador no entiende de subjetividades. La orden "t" va acompañada de un número entre 32 y 255, donde 32 sería equivalente a "Molto Moderato" (Muy lento) y 255 a "Prestissimo" (Rapidísimo). No daremos aquí una norma general sobre su uso, sino que recomendamos encarecidamente que experimenteis con él al introducir vuestras composiciones hasta que os guste lo que se oiga.

Con esto acabamos de analizar lo más importantes comandos de la sentencia PLAY (ya hemos dicho que no con todos). Como colofón, recomendamos que escribas al programa que damos en la figura 2, que interpreta un espiritual negro seguido del famoso "Oh, Susana", y que experimenteis con él, cambiando notas y comandos, hasta que no se parezca en nada al original.

Clasificado x: sólo para entendidos

Para terminar este artículo introductorio, vamos a hacer un repaso de las otras posibilidades que nos da la conjunción del MSX con el circuito AY—3—8910, a través del comando SOUND.

Este "chip" está estructurado como se puede ver en la figura 3. En él podemos ver claramente 4 bloque: 3 canales (A, B, y C) de sonido y un canal de ruido, todos los cuales van a unirse en un mezclador que habilita o inhabilita cada uno de los canales y lo mezcla con ruido si así se lo dijésemos. Existe también unos generadores de "envolvente de amplitud" y otra serie de bloques anejos de utilidades varias. Vamos a explicar cada uno de estas cosas.

Los *CANALES DE SONIDO*. Están formados de una especie de memoria (los registros), en los que se guarda un número que tiene íntima relación con el tono del sonido. Cada canal tiene dos registros, uno de 8 bits (1's ó 0's) y otro de 4 bits, que conjuntamente dan el período (inverso de la frecuencia) del sonido. Cuanto mayores sean estos números, más grave será el sonido. De los 12 bits implicados, los 4 que van "sueltos" son los que más influyen.

EL CANAL DEL RUIDO. Sólo dispone de 5 bits en un registro, el 6, que marcan la frecuencia del zumbido, al igual que los 12 bits de los canales de sonido fijaban la frecuencia del tono.

EL MEZCLADOR (de hecho son tres, uno para cada canal de sonido), pone en marcha o para el sonido generado por cada canal, además de mezclar cada canal con el ruido, si se le indicase así. El conjunto formado por los tres mezcladores lo controla el registro 7,

que está estructurado como aparece en la figura 4. La presencia de un "1" inhabilita la opción que represente, el "0" la habilita.

Los CONTROLES DE AMPLITUD fijan el volumen de cada canal. Los controlan los registros 8 a 10, de sólo 5 bits, de los cuales el de más peso habilita la posibilidad de un volumen variable controlado por el control de envolvente.

EL CONTROL DE ENVOLVENTE dispone de tres registro para su manejo. Los registros 11 y 12 controlan el ciclo de repetición de la envolvente de volumen, mientras que el registro 13 selecciona el tipo de envolvente. Los tipos de envolvente pueden encontrarlos en la figura 5.

EL CONVERTIDOR A/D es el que verdaderamente genera el sonido que oímos, dando por separado cada uno de los tres canales, que en los MSX se mezclan entre sí y con la señal de TV para enviarlo a nuestro receptor.

Los símbolos utilizados en la creación del sonido son similares a los escritos en una partitura.

Para poder usar directamente cada una de las opciones que da el generador de sonido, disponemos del comando del BASIC "SOUND". Este comando toma dos valores separados por una coma de cuales el primero es el número de registro que queremos alterar, y el segundo el valor que le queremos dar. Como ejemplo de su manejo, escriba el programa de la figura 6, y, por supuesto, experimente con él.

Esta explicación del *Generador Programable de Sonidos* ha sido muy escueta, pero es válida como introducción rápida a su manejo. Volvemos a recomendar la experiencia como madre de la ciencia, pero anunciamos ya un futuro artículo sobre este circuito integrado en un próximo número, con más datos técnicos y ejemplos sobre su manejo.

Conclusión

Como despedida, diremos que el sonido en el MSX tiene unas posibilidades casi infinitas y que, por más que exploremos, nunca llegaremos a agotar. Por enésima vez, recomendamos la experimentación como fuente principal de conocimientos. □

```
10 REM Un efecto curioso y sencillo usando el AY-3-8910
20 SOUND 0,129 :REM ajustamos el tono
30 SOUND 1,8   :REM del canal A
40 SOUND 6,25  :REM periodo de ruido
50 SOUND 8,31  :REM amplitud controlada por envolvente
60 SOUND 11,53 :REM ajuste del periodo
70 SOUND 12,25 :REM de la envolvente
80 SOUND 13,8  :REM envolvente en diente de sierra
90 SOUND 7,&B11111110
100 REM el canal a tiene un sonido sin ruido
110 FOR I=1 TO 4000:NEXT I
120 REM esperamos un poco
130 SOUND 7,&B11110110
140 REM ahora suena con ruido
150 FOR I=1 TO 4000:NEXT I
160 SOUND 2,132:SOUND 3,0:SOUND 7,&B11111100
170 FOR I=1 TO 15: REM vamos aumentando poco a poco el volumen del canal B190 FO
R J=1 TO 150
180 SOUND 9,I
190 FOR J=1 TO 150
230 NEXT J
240 NEXT I
250 FOR I=1000 TO 1 STEP -1:NEXT I
260 FOR I=15 TO 1 STEP -1:REM y ahora lo bajamos poco a poco
270 SOUND 9,I
280 FOR J=1 TO 150:NEXT J
290 NEXT I
300 FOR J=1 TO 1000:NEXT J
310 SOUND 8,0: REM con esto apagamos
320 SOUND 9,0: REM todo el sonido
```

Bienvenidos a esta nueva sección, la cual estará abierta a todos aquellos que deseen anunciarse, abriendo un camino por el que podamos transmitir, tanto nuestras ideas, como nuestros deseos de comprar, vender o intercambiar los diversos productos que pueblan el mercado del MSX en estos momentos, que aunque no sea diverso resulta muy interesante.
Los anuncios publicados en esta sección serán gratuitos y perseguiremos aquellos que por su carácter sean dudosos o tengan afán de lucro.





• VENDO Spectravideo SV-328, superexpander 605B (2 × 320Kb), 80 columnas, monitor, en garantía y programas diversos por 300.000 pts., sin monitor por 270.000 pts., y sin los programas por 200.000 pts. Llamar a Ignacio García al Telf. 254 37 19 de Madrid.

• Deseo INTERCAMBIAR programas y experiencias de SV-318 y SV-328. Escribid a José Jorge Vaz, Travesía de Vigo, 22, 3-B. Vigo 6. Pontevedra.

• VENDO ordenador Spectravideo SV-728 (MSX). Llamar a José Joaquín por las mañanas al Tel. 705 88 31 de Madrid.

• Dentista VENDE Spectravideo, unidad doble de diskettes nuevos, un lote de programas de aplicación (Wordstar, multiplan, etc.) y un lote de lenguajes de programación (MBASIC, COBOL, etc.). En garantía, por 300.000 pts., al contado. Llamar o escribir a Julio Montolio Salvador, c/ Fernando el Católico, 40, 1-C. 50009 Zaragoza. Telf. (976) 45 08 38.

• VENDO Spectravideo SV-328 con superexpander, ampliación de 64K, controlador de discos, 2 unidades de discos, tarjeta de 80 columnas, interface y cable de impresora, monitor Datalec (de fósforo verde), CP/M, DBASE 2 y manuales, por 250.000 pts. Llamar a José Closas Compte al tel. (93) 239 39 03 de Barcelona.

• VENDO Spectravideo SV-328, con superexpander (1 disco), libros y programas nuevos, discos de limpieza, CP/M y BASIC de disco, Assembler, comprado en julio- 84. Todo por 150.000 pts. Llamar al tel. 338 03 94 de Barcelona, preguntar por Manuel.

• VENDO Sony HIT-BIT HB-55/ 75P, 16K ampliable a 68K con cassette y garantía de 1 año. Todo por 45.000 pts. Llamar a Salvador Puig Oliva al tel. 384 39 63 de Barcelona.

• Club de usuarios de MSX. Para contactar y unirse a nosotros, dirígete a Isabel Linares.
Los Chopos, 34. Las Gabias (Granada).

• Formación de un club de usuarios en Gerona para intercambiar, información, contactos, etc. Interesados escribir a Josep Miquel, c/ Río Ser, 4, 3-B. 17003 Gerona.





Papel, piedra o tijera



¿Quién no habrá intercambiado cromos alguna vez? Más de uno se ha visto en la situación de tener un álbum de cromos y ver que los cinco que le faltan par completar la ansiada colección los tiene un amigo. Había formas y "formas" de cambiarlos, una era el popular *"Pares o Nones"*, donde cada uno elegía una u otra opción y en base a ella, el que ganaba se llevaba un cromo, si perdía se encontraba en la desagradable situación de tener que dar alguno. Este juego, para el cuál presentamos aquí su programa, es una variante del anterior, pero algo más divertida. Se juega de la misma manera que "Pares o Nones" pero en lugar de contar los dedos que cada uno sacaba y sumarlos, se comprueba si ocurre uno de los casos siguientes: la tijera corta el papel, la piedra puede romper la tijera y el papel envuelve la piedra. Estos son los posibles casos que se pueden obtene. El jugador que saque "piedra" (el puño cerrado) frente al que saque "tijera" (dos dedos en forma de V) gana, el que saque "papel" (la mano abierta) frente al que saque "tijera" perdía, y así con el resto de las opciones que no voy a describir ya que son elementales, además de que le quitaríamos algo de interés.

El programa está realizado en BASIC, donde resaltaremos la utilización de varias instrucciones como son **KEY OFF** y **SCREEN**. La primera hace desaparecer la última línea de la pantalla, donde se definen los comandos de las teclas de función y el segundo comando prepara las dimensiones de la pantalla en lo que se refiere a los sprites, ya que como hemos visto, existen cuatro tamaños posibles que se definen con la sentencia **SCREEN**.

Hay que destacar el tamaño de los sprites, siendo estos de 16 x 16 pixels aumentados y que la representación gráfica de los tres elementos están muy bien conseguida.

Al ejecutar el programa, aparecerán dos figuras aleatorias que habrá que parar y enfrentar. El jugador A controlará su figura con la barra espaciadora y el B con la techa **RETURN**, mientras que el ordenador será el que decida quien de los dos ha ganado.

```
10 REM Papel, piedra o tijera
20 CLS:SCREEN 1,3:COLOR 11,1,1
30 KEY OFF:DIM J(2,2)
40 FOR I=0 TO 2:FOR J=0 TO 2
```

```
50 READ J(I,J):NEXT:NEXT
60 FOR J=0 TO 2:T$=""
70 FOR I=1 TO 32:READ D$
80 T$=T$+CHR$(VAL("&h"+D$))
90  NEXT:SPRITE$(J)=T$
100 NEXT
110 ON STRIG GOSUB 240:STRIG(0) ON
120 LOCATE 3,3:PRINT "PAPEL, PIEDRA O TIJERA"
130 LOCATE 4,6:PRINT"JUGADOR          JUGADOR"
140 LOCATE 7,8:PRINT "a                 b"
150 A1$="l64t255n"+STR$(A*5+50)
160 A2$="l64t255n"+STR$(B*5+80)
170 PLAY A1$,A2$
180 PUT SPRITE 0,(60,80),2,A
190 PUT SPRITE 1,(160,80),2,B
200 A=A+1:IF A>2 THEN A=0
210 B=B+1:IF B>2 THEN B=0
220 FOR I=0 TO 30:NEXT
230 GOTO 170
240 STRIG(0) OFF:A=INT(RND(-TIME)*3)
250 PUT SPRITE 0,(60,80),2,A
260 A1$="l64t255n"+STR$(B*5+80)
270 PLAY A1$
280 PUT SPRITE 1,(160,80),2,B
290 B=B+1:IF B>2 THEN B=0
300 FOR I=0 TO 30:NEXT
310 D$=INKEY$:IF D$<>CHR$(13) THEN 260
320 B=INT(RND(-TIME)*3)
330 PUT SPRITE 1,(160,80),2,B
340 CA=5:CB=CA
350 IF J(A,B)=0 THEN M$="Empate"
360 IF J(A,B)=1 THEN M$="Gana a":CB=6
370 IF J(A,B)=2 THEN M$="Gana b":CA=6
380 PUT SPRITE 0,(60,80),CA,A
390 PUT SPRITE 1,(160,80),CB,B
400 LOCATE 11,18:PRINT M$
410 D$=INKEY$:IF D$<>CHR$(13) THEN 400
420 LOCATE 11,18:PRINT "       "
430 STRIG(0) ON:RETURN
440 DATA 0,1,2,2,0,1,1,2,0
450 DATA 0,0,1,3,f,3f,7f,7f
460 DATA 7b,77,7f,3f,1f,f,3,0
470 DATA 0,78,fc,fe,7f,bb,fd,ff
480 DATA ff,f7,fb,ff,fe,fc,f0,0
490 DATA 0,0,0,0,0,0,0,0
500 DATA ff,3f,0,1,2,2,2,1
510 DATA 80,80,c0,c0,c0,c0,ce,d1
520 DATA f1,fe,c0,c0,40,40,40,80
530 DATA 3f,20,20,20,20,20,20,20
540 DATA 20,20,20,20,20,20,20,3f
550 DATA f0,28,24,22,3e,2,2,2
560 DATA 2,2,2,2,2,2,2,fe
```



DEFINICION DE SPRITES

Soy un feliz poseedor de un GOLDSTAR FC-200, pero mis conocimientos acerca del lenguaje BASIC no son muy profundos. Por esta razón quisiera que me explicaran, ¿qué es un *sprite*?

Alberto Giménez
Burgos

Los *sprites* son caracteres definidos por el usuario que se pueden visualizar en la pantalla sin que estos afecten el contorno. Esto es debido a que los *sprites* se van definiendo en los distintos planos que forman una pantalla. Este mismo concepto es el que hace que un gráfico se pueda esconder detrás de otro sin distorsionar las figuras.

Existen cuatro tamaños de *sprites*, desde el de 8 × 8 *pixels* normales hasta el 16 × 16 *pixels* aumentado, esto se define con la sentencia SCREEN, al comienzo del programa.

POSIBILIDAD DE INTERCONECTAR MSX

Somos un grupo de amigos que poseemos ordenadores MSX, con la siguiente duda. ¿Es posible la creación de una red local de transmisión de datos para ordenadores MSX?

Alberto Sánchez
Madrid

Las innovaciones en el MSX están al orden del día. Es más, en las páginas de noticias de este número, hablamos de la presentación de una red local, diseñada por Spectravideo con la posibilidad de poder conectar hasta 32 ordenadores MSX, con el único requisito de que estos posean un mínimo de 32K de memoria. De esta forma podremos trabajar conectados entre sí o independientemente del ordenador central.



DUDA METODICA

La pega la encuentro en el microprocesador Z80, y aunque sea de fiabilidad más que probada, ¿no se está haciendo viejo? por que actualmente el mercado está saturado de procesadores de 8 *bits*. Pienso que el futuro de los ordenadores personales están en máquinas pequeñas con procesadores de 16 *bits*.

José Luis López
Granada

Efectivamente, en el número anterior hicimos referencia al futuro de los ordenadores MSX, y auguramos un buen porvenir. Además tienes razón en cuanto a los microprocesadores de 16 *bits*, ya que es la solución ideal a la capacidad (velocidad de proceso y memoria) de estos ordenadores. De cualquier manera tenemos noticias de que ya se están fabricando la nueva ola de ordenadores MSX de 16 *bits*, para los cuales han introducido en lugar del Z80, el Z800, de esta forma la compatibilidad entre los nuevos ordenadores y los que ya llevan una temporada en el mercado permanecerá inalterable.

DIRECTOR:
Juan Arencibia
COORDINADOR EDITORIAL:
Emiliano Juárez
REDACCION:
Fernando García, Santiago Gala, Ricardo García, Teresa Aranda, Francisco Mancera.
DISEÑO:
Ricardo Segura y Benito Gil.
Editada por
PUBLINFORMATICA S.A.
PRESIDENTE:
Fernando Bolín.
DIRECTOR EDITORIAL:
Norberto Gallego.
Administración:
PUBLINFORMATICA
GERENTE DE CIRCULACION Y VENTAS:
Luis Carrero.
PRODUCCION:
Miguel Onieva.
DIRECTOR DE MARKETING:
Antonio Gonzalez.
SERVICIO AL CLIENTE:
Julia Gonzalez.
Tel. 733 79 69
ADMINISTRACION:
Miguel Atance, Antonio Torres.
JEFE DE PUBLICIDAD:
María José Martín.
DIRECCION Y REDACCION:
C/ Bravo Murillo, 377 - 5° A
Tel. 733 74 13
28020 MADRID
PUBLICIDAD Y ADMINISTRACCION:
C/ Bravo Murillo 377 - 3° E
Tel. 733 96 62-96
Publicidad en Madrid:
Fernando Hernando
Publicidad en Barcelona:
Olga Martorell
C/ Pelayo, 12
Tel. (93) 301 47 00 Ext. 27-28.
08001 BARCELONA
Deposito Legal: M. 16.755-1985
Impreso en Héroes S.A.
C/ Torrelara, 8. 28016-MADRID.
Distribuye:
S.G.E.L. Avda. Valdelaparra s/n.
Alcobendas (Madrid).